Exhibit 1

microunity
MicroUnity Systems Engineering, Inc.

REDACTED

Mark Birrittella
Development Building
Cray Research, Inc.
900 Lowater Road
Chippewa Falls, WI 54729

Re: Technology Review Presentation Materials

Dear Mark:

A copy of the presentation materials from the REDACTED technology review is enclosed for use by Cray Research, Inc. in accordance with that certain License Agreement between MicroUnity Systems Engineering, Inc. and Cray Research Inc. dated REDACTED. Under this agreement, Cray has an obligation to protect information disclosed pursuant to the agreement which is "in written, graphic, machine readable or other tangible form and is conspicuously marked 'Confidential', 'Proprietary' or in some other manner to indicate its confidential nature." The quarterly review presentation materials are confidential information.

Please contact me upon your receipt of this letter to verify proper delivery of the materials. I may be reached at (408) 734-8100.

Sincerely,

T. B. Robinson

Tim Robinson
Director of Systems Engineering

MU 0020319

Enclosure: Copy of the REDACTED technology review presentation materials

cc: John Moussouris, MicroUnity Systems Engineering, Inc.

CONFIDENTIAL

255 CASPIAN DRIVE, SUNNYVALE, CALIFORNIA 94089-1015 TEL: 408 734-8100 FAX: 408 734-8136

Agenda for the Cray Research and MicroUnity Review

REDACTED

Tuesday

2.00 PM Introduction

2.15 PM Process Status
Paul Poenisch

4.00 PM Split for Business Meeting Discussion

Wednesday

8.00 AM Architecture Update
Craig Hansen
Tom Karzes

10.00 AM Circuits Update
Bill Herndon
Geert Rosseel

11.00 AM Euterpe Implementation
Geert Rosseel
Tim Robinson

12.00 PM Lunch - Discussion

1.00 PM Meeting Concludes

MU 0020320

CONFIDENTIAL

MicroUnity I.C. Process Status

Agenda

- Introduction - process overview
- Historical perspective
- Current facility and equipment status
- Process status
- Current Device Status
- Documentation
- Summary

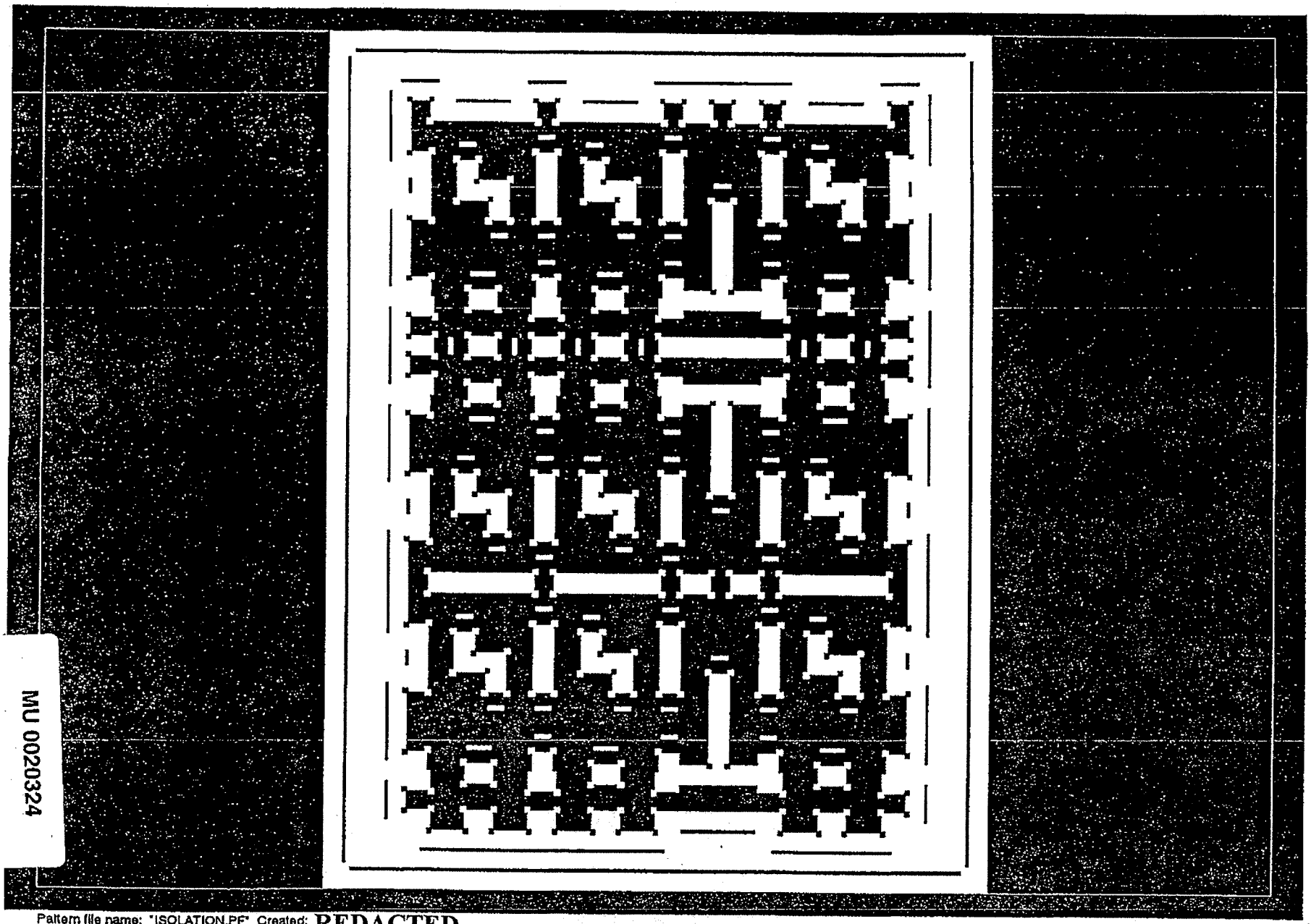
MicroUnity I.C. Process Status

Key Features of MOBI MOS 1

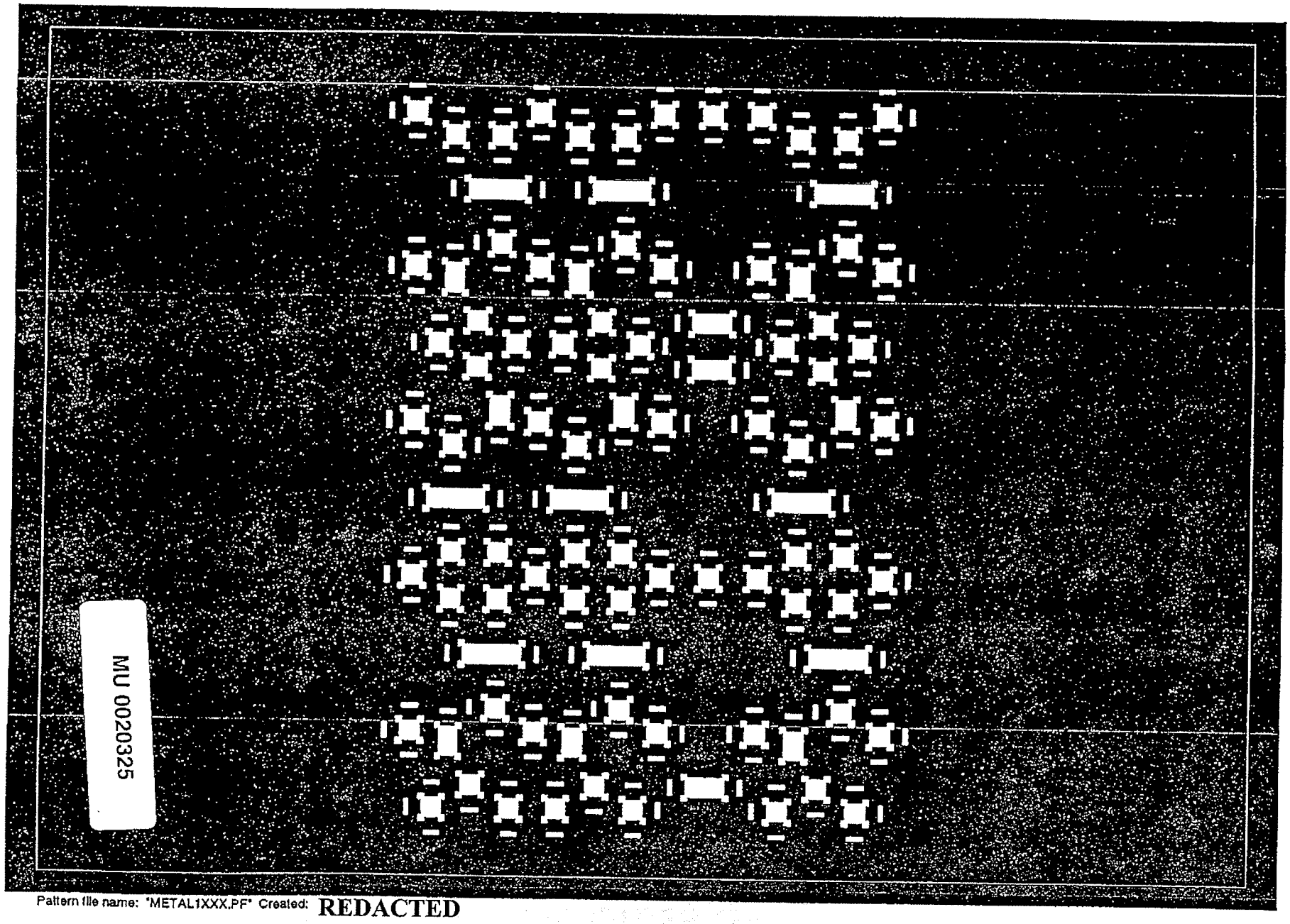
- 0.5 micron line and space on all layers.
- Advanced, non phase shifting, reticles.
- Maximum non-planarity at photomasking and metal deposition of < 0.15 microns.
- Four routing layers of metal, top two are air bridged.
- Symmetric PMOS and NMOS transistors.
- F_t of bipolar transistors > 40 GHz.

Key Features of MOBI MOS 1 (continued)

- Package consists of die, space transformer and TAB.
- Metallization is inherently electromigration resistant.

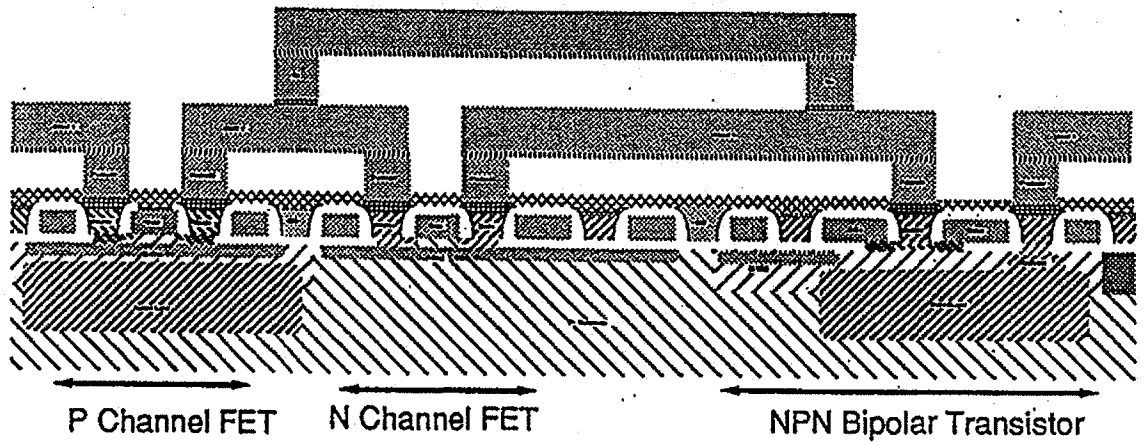


Pattern file name: "ISOLATION.PF" Created: REDACTED

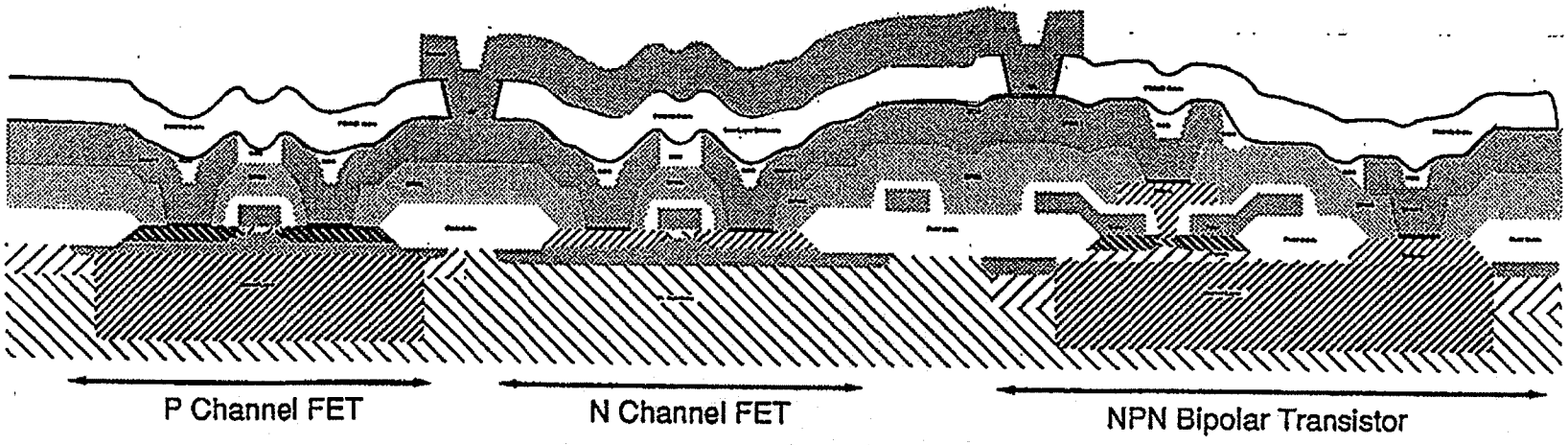


Pattern file name: "METAL1XXX.PF" Created: REDACTED

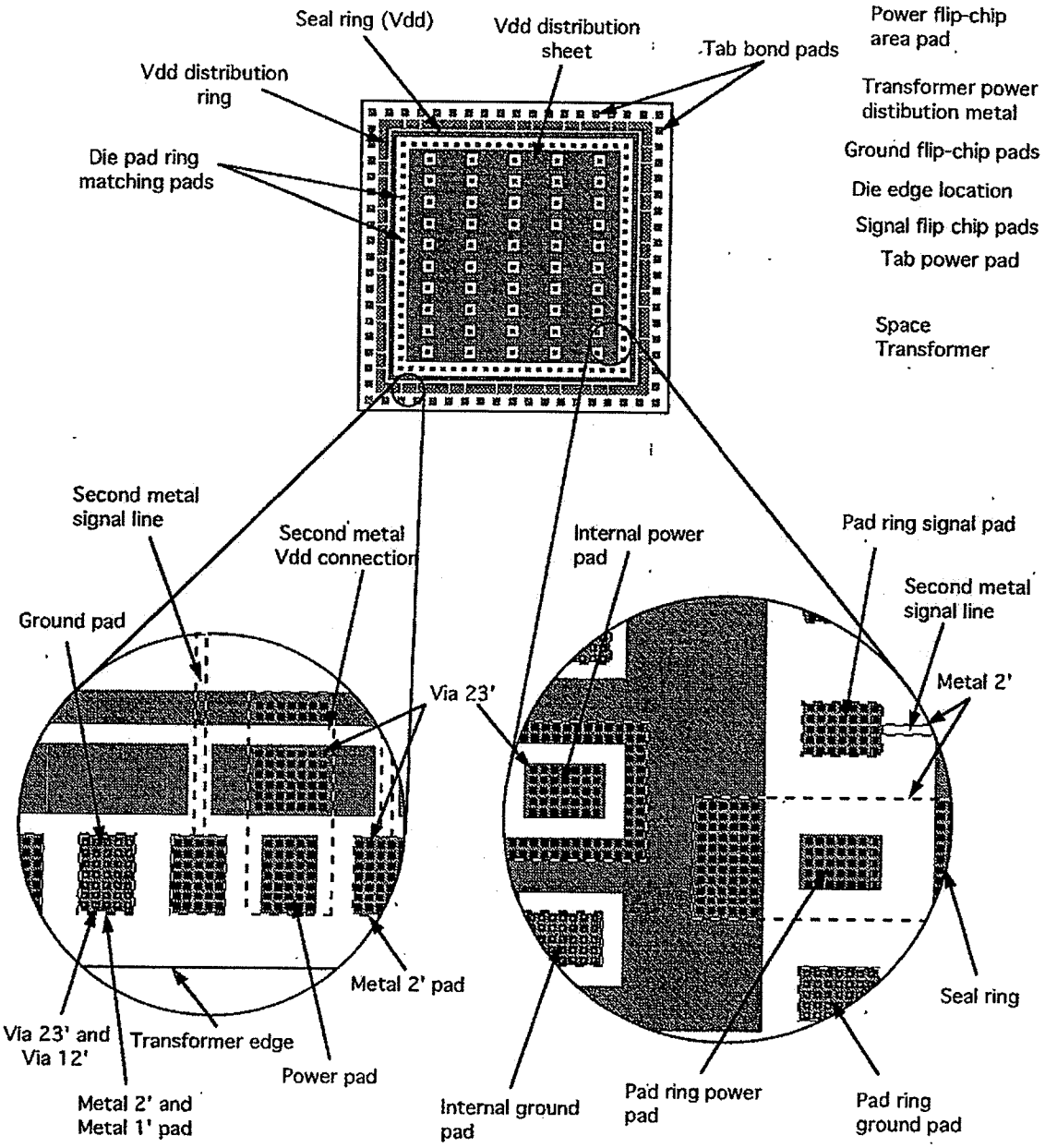
MicroUnity BiCMOS Process Cross Section



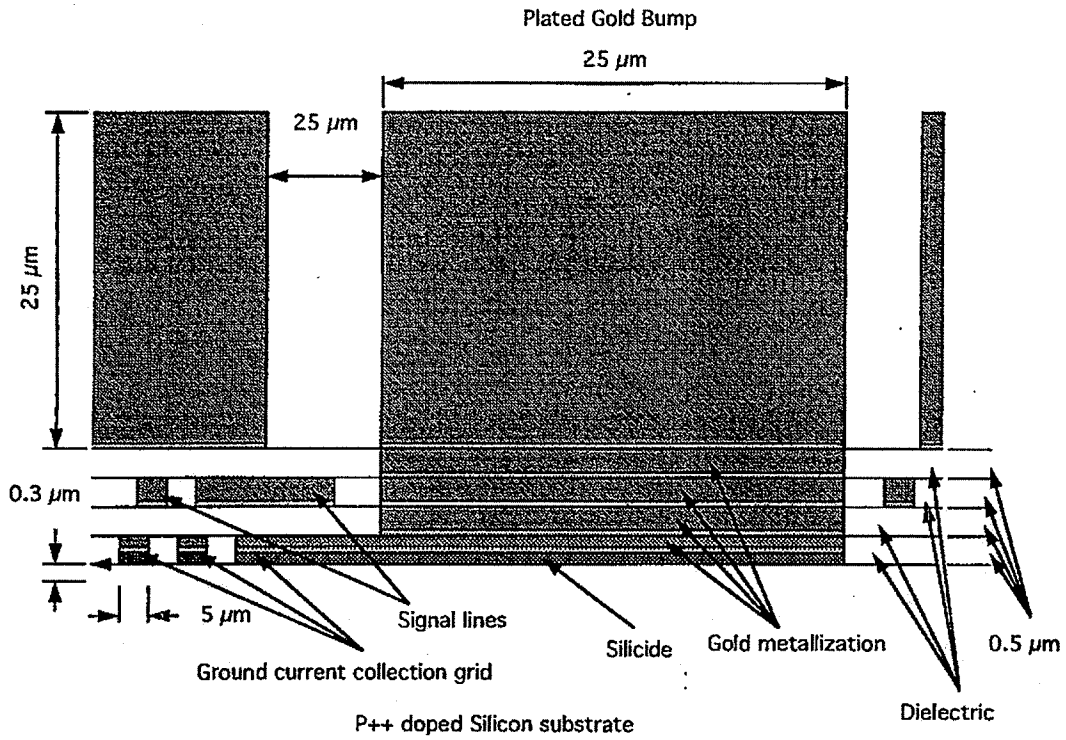
Conventional BiCMOS Process Cross Section

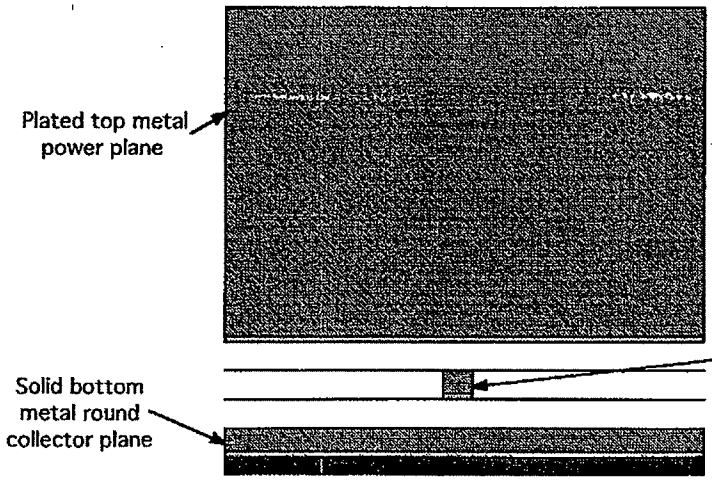


Space Transformer Layout



Space Transformer Structure





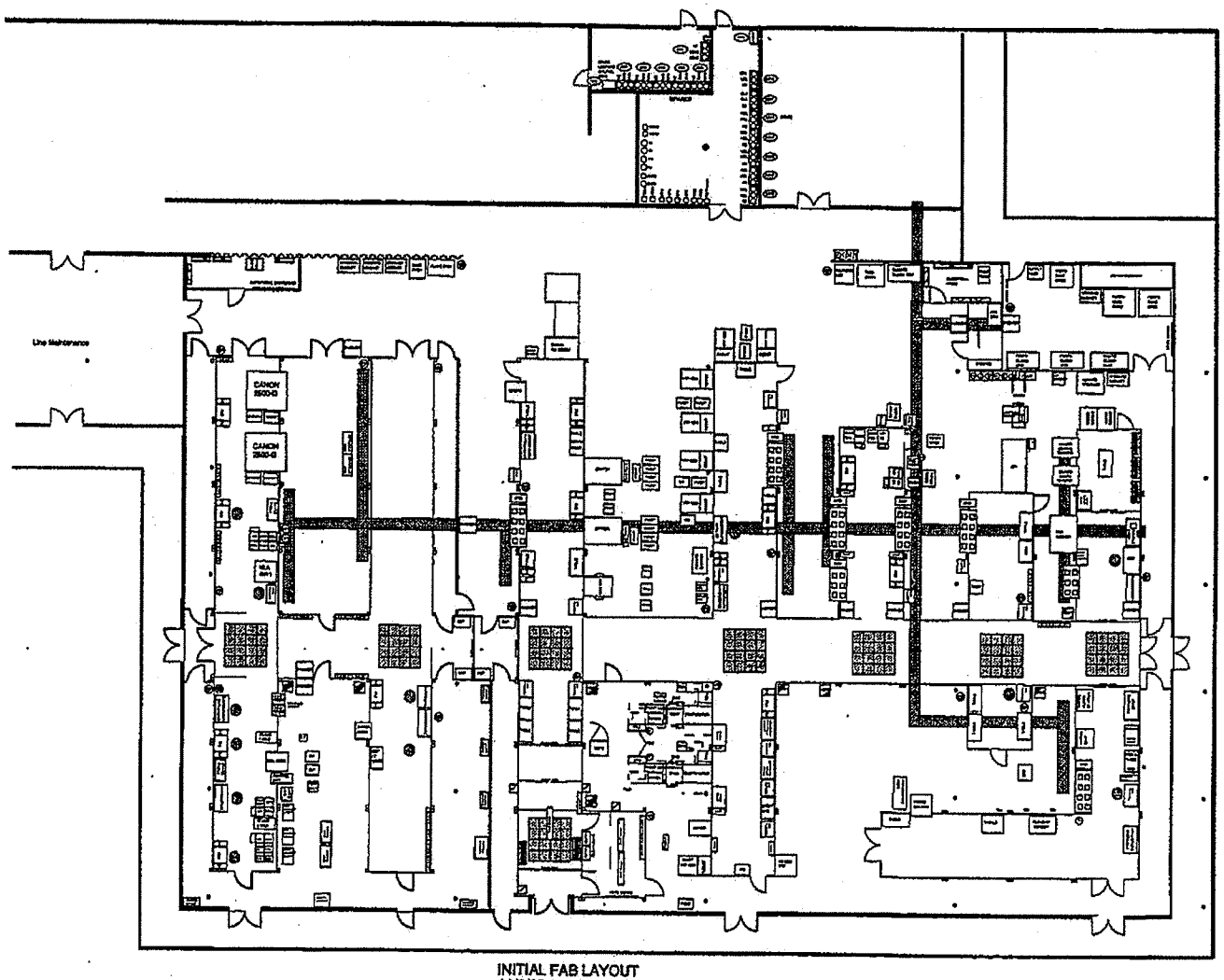
Signal Line Impedance Control

MU 0020328

CONFIDENTIAL

MicroUnity Systems Engineering, Inc.

Vendor	Digital	Fujitsu	HP	IBM	IBM	Intel	TI	TI	MicroUnity
Process Name	CMOS-5	CS-50	CMOS-14	CMOS-5S	CMOS-5X	"0.6 micron"	EPIC-2BE	EPIC-3	MOBIMOS
Example Product	21064A	Sparc-2	PA-7200	PPC 620	PPC 601+	P54C	S'Sparc	MVP	Calliope
First Production	3Q94	1Q94	4Q94	4Q94	4Q94	1Q94	2Q94	3Q94	**4Q94**
Supply Voltage	3.3 V	3.3 V	4.4 V	3.3 V	2.5 V	3.3 V	4.8 V	3.3 V	**3.3 V**
BiCMOS?	no	no	no	no	no	**yes**	**yes**	**opt**	**yes**
Gate Length: Drawn (microns)	0.50	0.50	0.55	**0.50**	**0.50**	0.50	0.60	0.55	**0.50**
Gate Length: Effective (microns)	0.37	0.45	0.38	**0.39**	**0.25**	0.37	0.50	0.47	**0.35**
Gate Oxide Thickness (angstroms)	90	110	120	**90**	**70**	80	120	90	**108**
No. of Metal Layers	4	3 - 4	3	**5**	**5**	4	3	3 - 4	**5**
Local Interconnect?	yes	no	no	**yes**	**yes**	no	yes	no	**yes**
Stacked Vias?	no	no	no	**yes**	**yes**	no	yes	yes	**yes**
M1 contacted pitch (microns)	1.5	2.1	1.8	**1.4**	**1.2**	1.4	2.0	1.8	**1.0**
M2 contacted pitch (microns)	1.8	2.1	1.8	**1.8**	**1.8**	1.7	2.0	1.8	**1.0**
M3 contacted pitch (microns)	5.0	2.1	2.4	**1.8**	**1.8**	1.7	2.6	2.4	**1.0**
M4 contacted pitch (microns)	5.0	210	-	**1.8**	**1.8**	3.5	-	4.0	**1.0**
Routing Index (square microns)	4.9	4.4	4.3	**2.7**	**2.5**	2.9	4.3	4.1	**0.8**

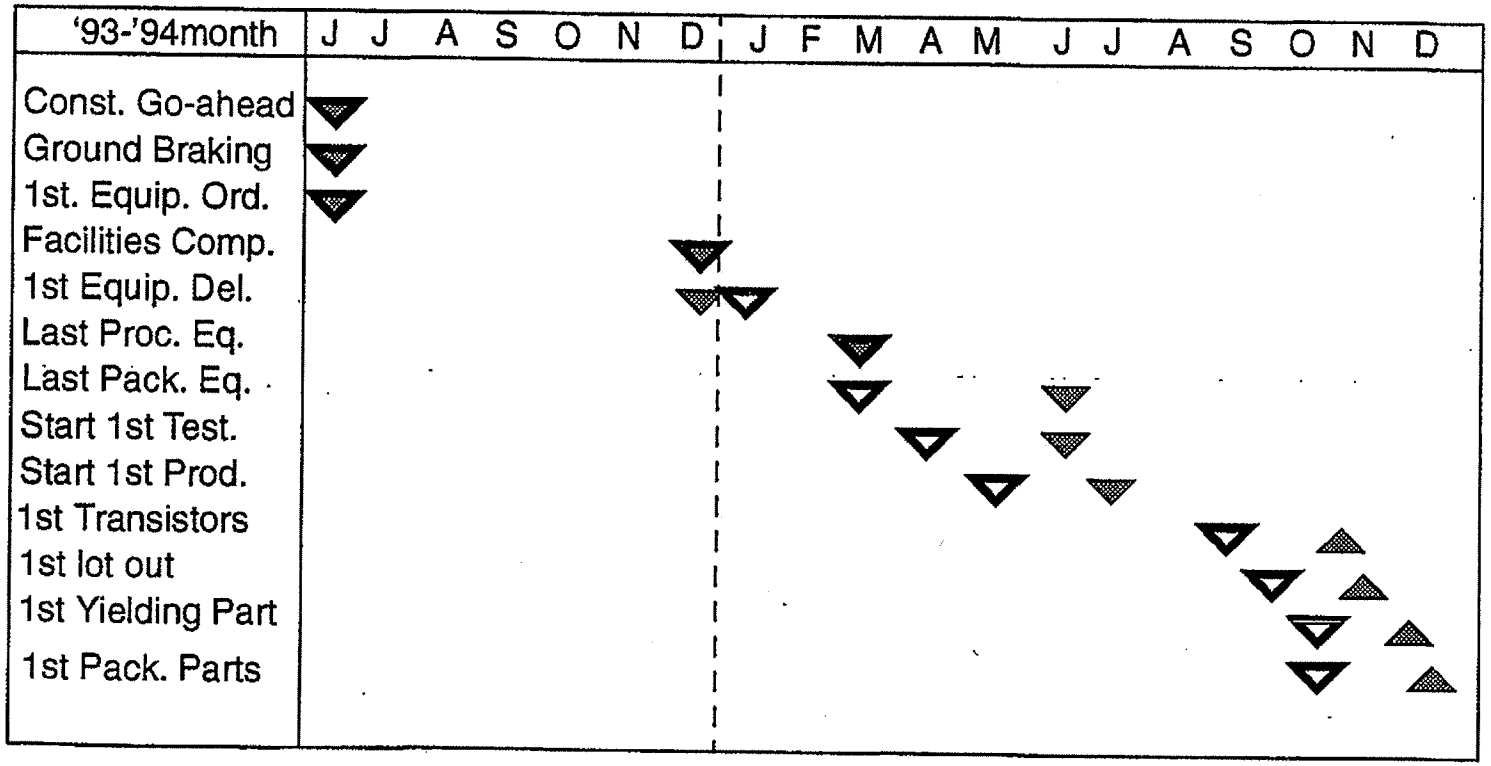


INITIAL FAB LAYOUT
11/9/93
REV. 4

MicroUnity I.C. Process Status

Historical perspective on MicroUnity's I.C. Fab

- Time line of events



MicroUnity I.C. Process Status

MicroUnity Fab current status

- Facilities
  - — The fab was designed to provide a cleanliness level of class 10 or better.

    Currently the fab is running below the class 1 level 95% of the time with occasional excursions to ~ class 10.
  - — Facilities are 95% built out, 100% by December.
  - — Temperature tracking +/- 0.25 F.
  - — Humidity tracking +/- 1% RH.

MicroUnity Fab current status (continued)

- Equipment
  - — Photomasking
    - One i-line stepper in production operation
    - One i-line stepper in qualification
    - Resist spin coat capacity adequate for pilot operations
    - One additional develop track on order.
  - — Etch/PECVD/Ion implant
    - Two plasma etch systems (ten chambers) and one PECVD system (5 chambers) are in production operation
    - One medium current implanter is in production operation.

MicroUnity Fab current status (continued)

- Equipment (continued)
  - — Metallization

    Two metal evaporators (six pockets each) are in production operation

    One lift-off tool is being characterized by engineering and one is waiting bring-up

    Two plating stations are up and running (three tanks each, one in use, one ready for fill).
  - — Diffusion and Epi

    Seven vertical diffusion tubes have processes up and running on them and, six have been released to production

MicroUnity Fab current status (continued)

- Equipment (continued)
  - — Diffusion and Epi (continued)

    One epi system is up and released to production for the thin epi layer, thick layer is in engineering evaluation.
  - — Packaging

    Wafer saw, sawed wafer cleaner, wafer mounting station and developmental flip-chip bonder have been released to production

    TAB bonder, and airbridge equipment are in engineering evaluation.

MicroUnity I.C. Process Status

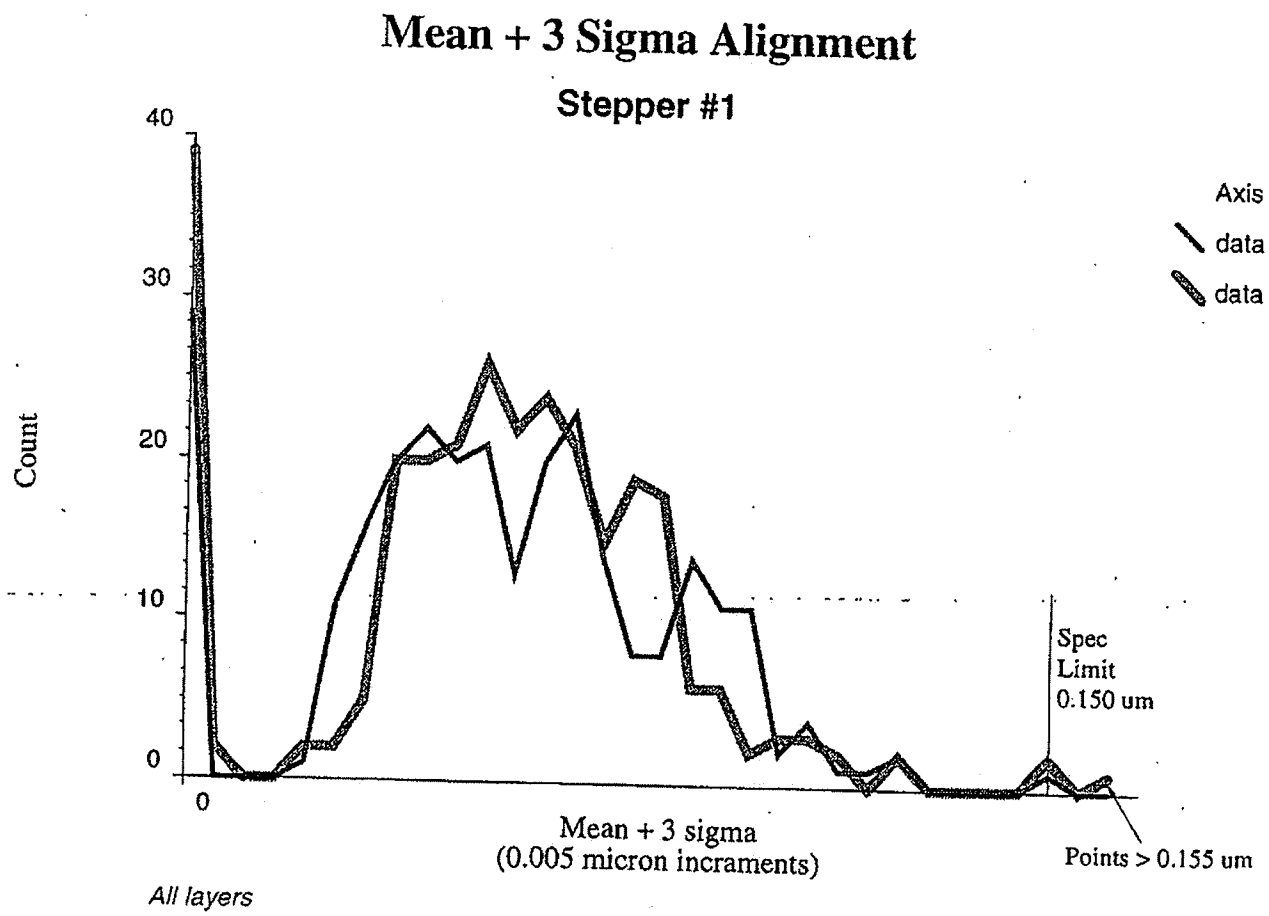
Process Status

- Transistors
  - — There are several critical alignments in the formation of the transistors (Bipolar and MOS). To date alignment on our single production stepper has been within the 3 sigma plus offset requirement, <0.15.
  - — "Poly wafflization" is the method chosen to maintain planarity and CD control at gate/base formation.

    CD is being gathered now and so far looks good, but more data is needed

    Layout was not adversely effected, SRAM cell is 22 sq. microns, the ECL atom is 96 sq. microns.

Alignment History



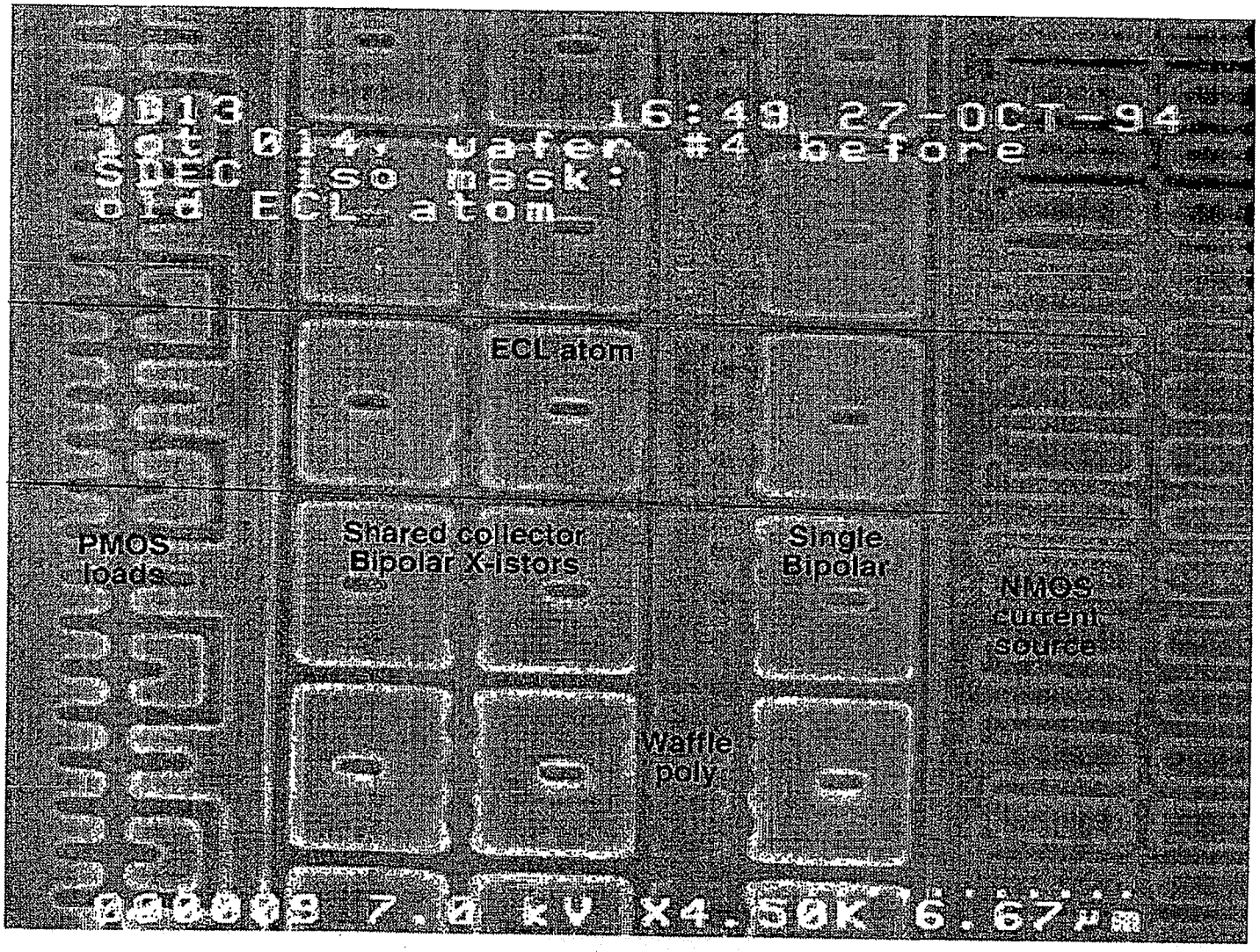
Process Status (continued)

- Transistors (continued)
  - — Source, drain, emitter and collector areas are extended vertically by polysilicon formation between the poly 1 features (SDEC)

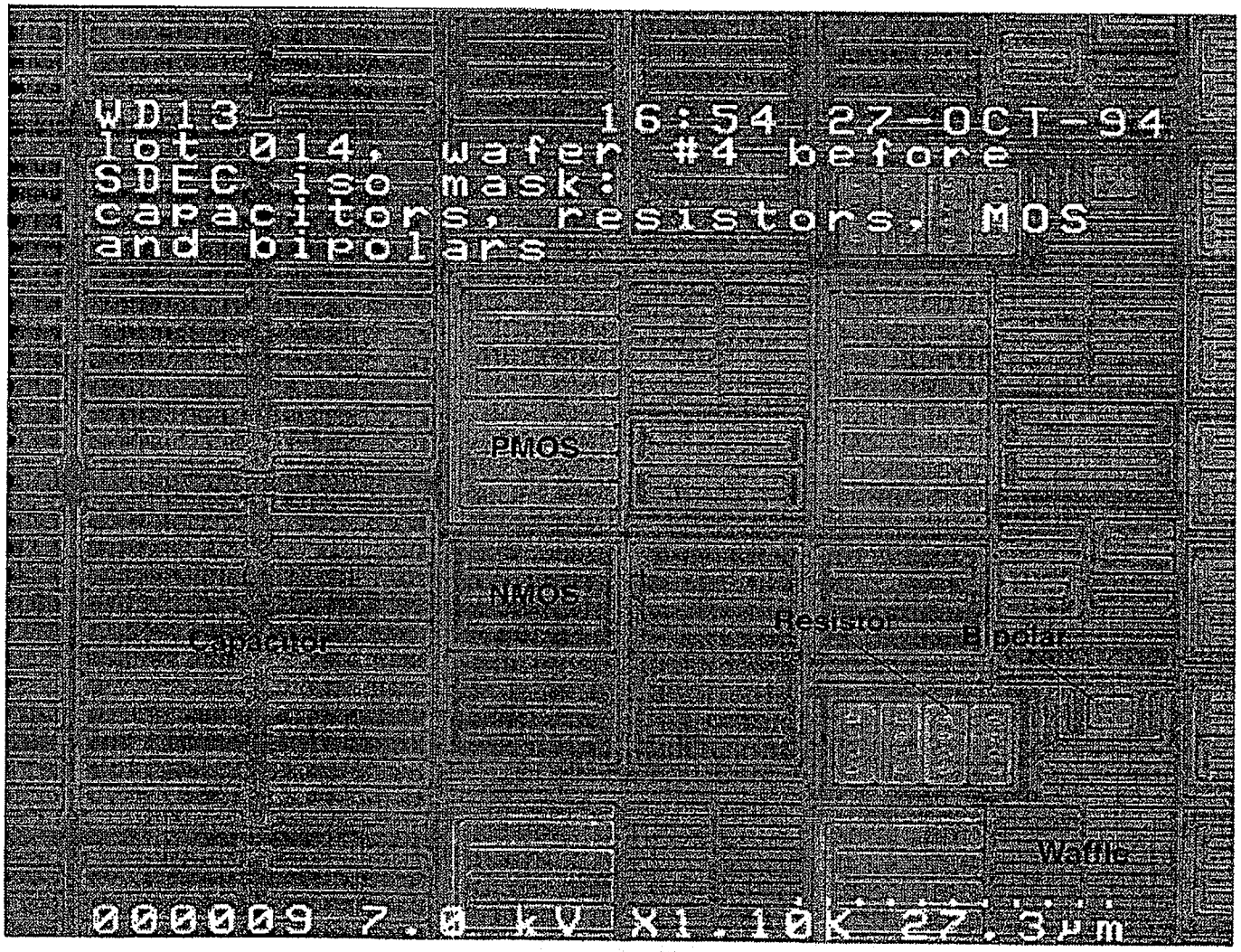
    SDEC formation is doable but more work is needed.
  - — Silicide used is $CoSi_2$

    Silicide appears to be stable with the metal system in use.
  - — No testable transistors have reached E-test yet, we expect this to occur within two weeks.

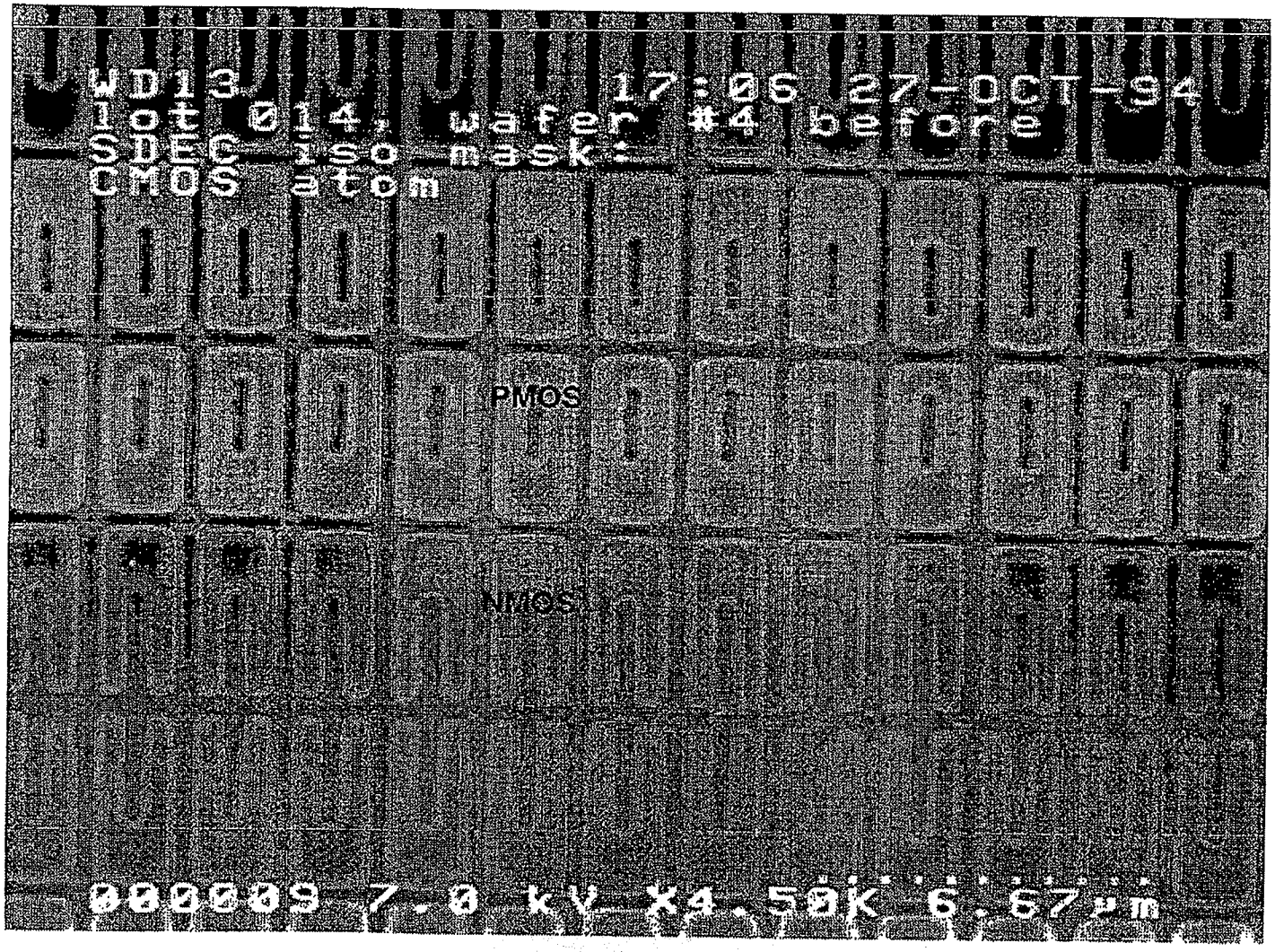
ECL Atoms



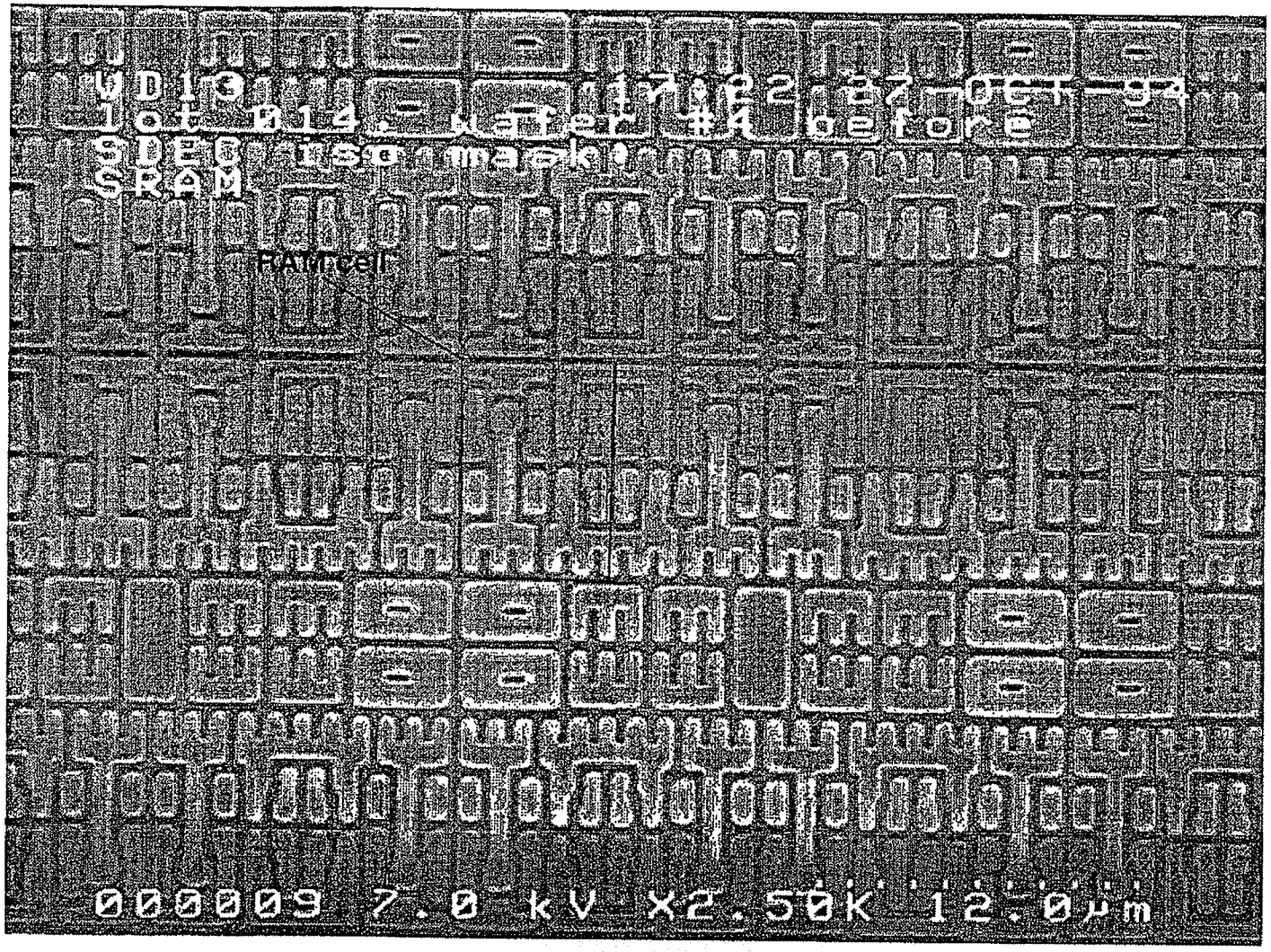
Analog Device Section



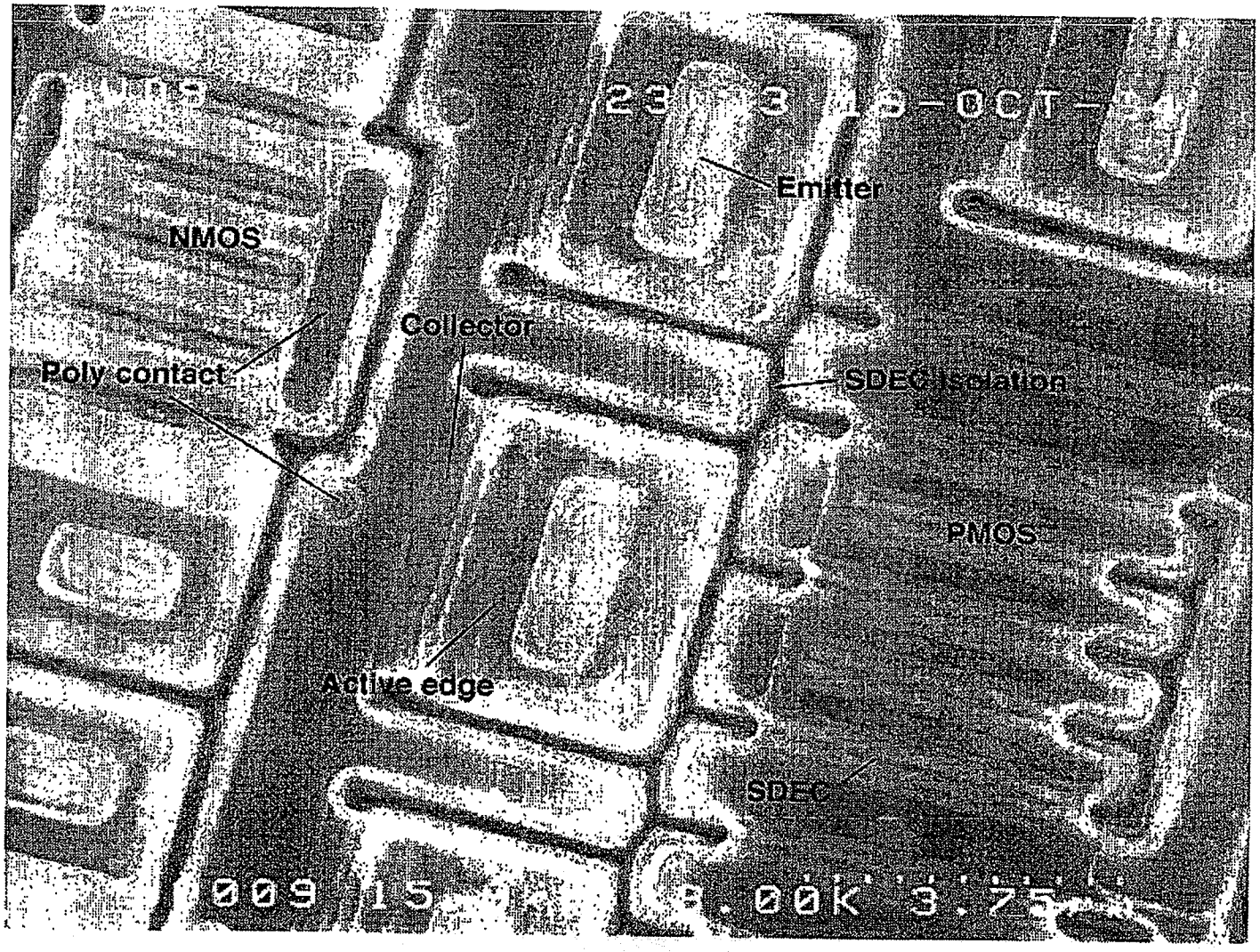
CMOS Atoms



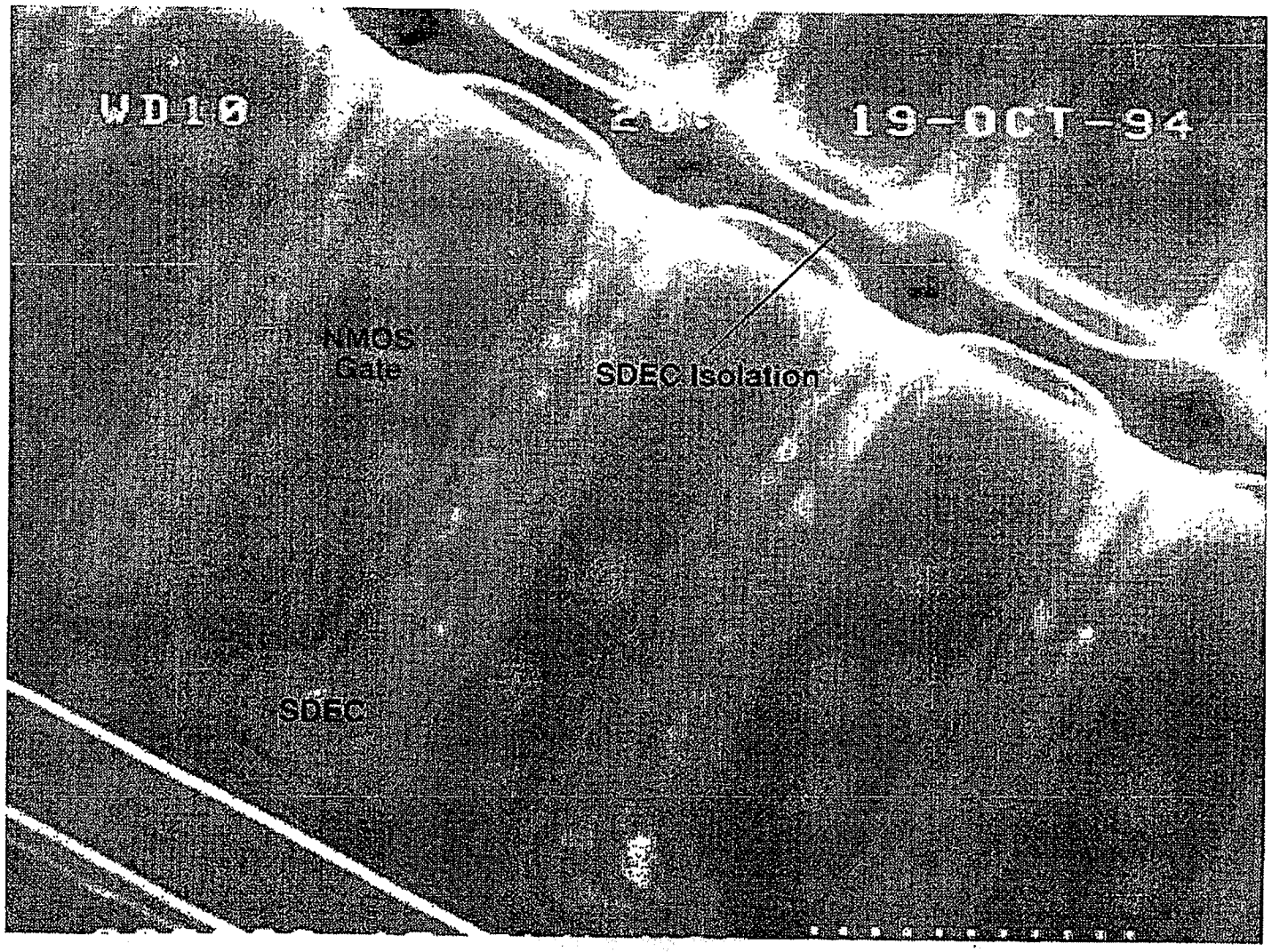
RAM Cell



SDEC, SDEC Isolation and $CoSi_2$



SDEC Isolation close-up



Process Status (continued)

- Metallization
  - There are two basic metal systems in use for the process: Ti/Pt/Au and Nb/Au

    The Nb/Au system is usable up to 400C* for extended times (longer than 1 hour).

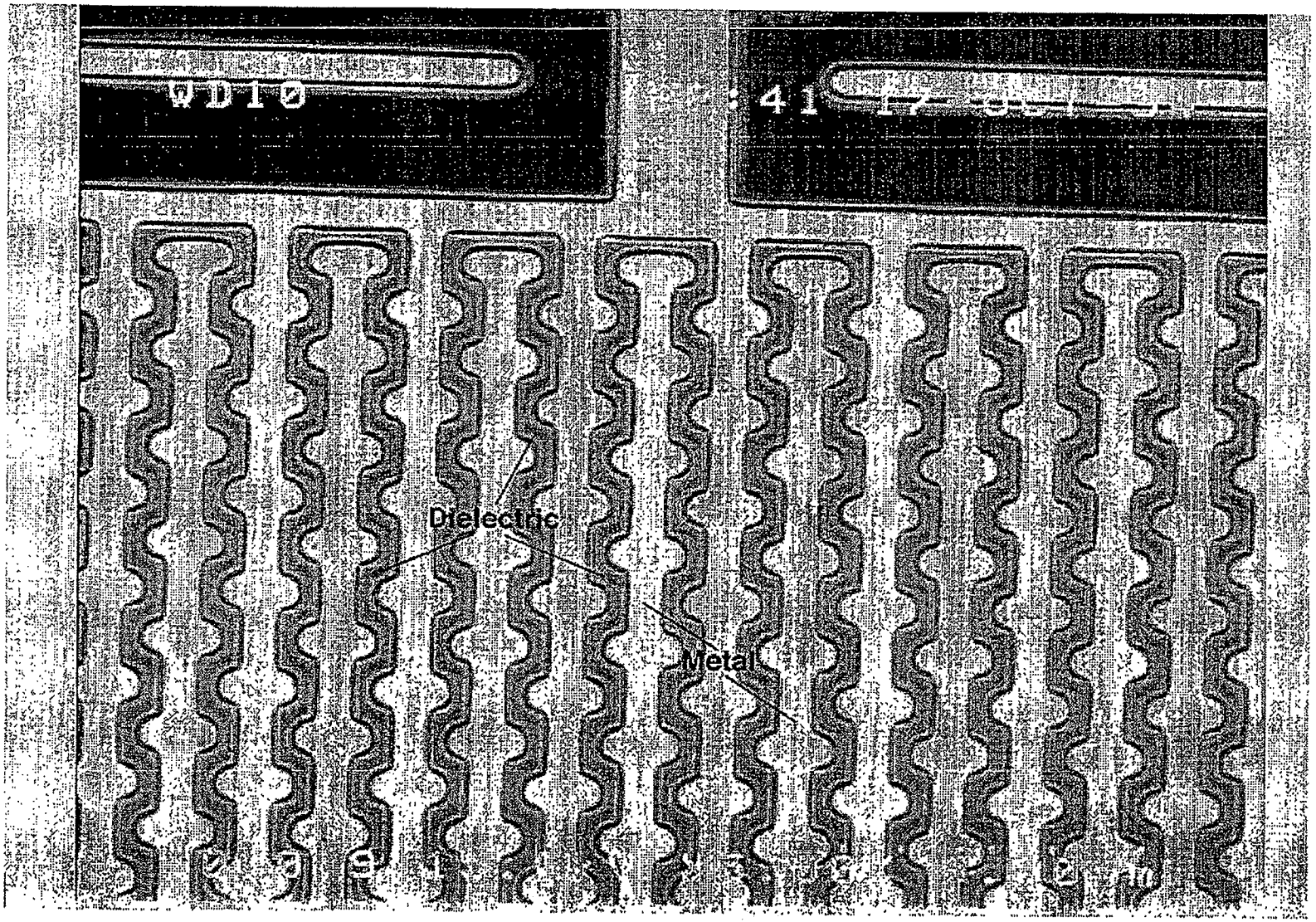
    The Ti/Pt/Au system is being used for the initial barrier between the metal systems and the transistors.
  - Lift-off is being used to pattern the metal layers

    Lift-off of the Ti/Pt/Au stack has been demonstrated

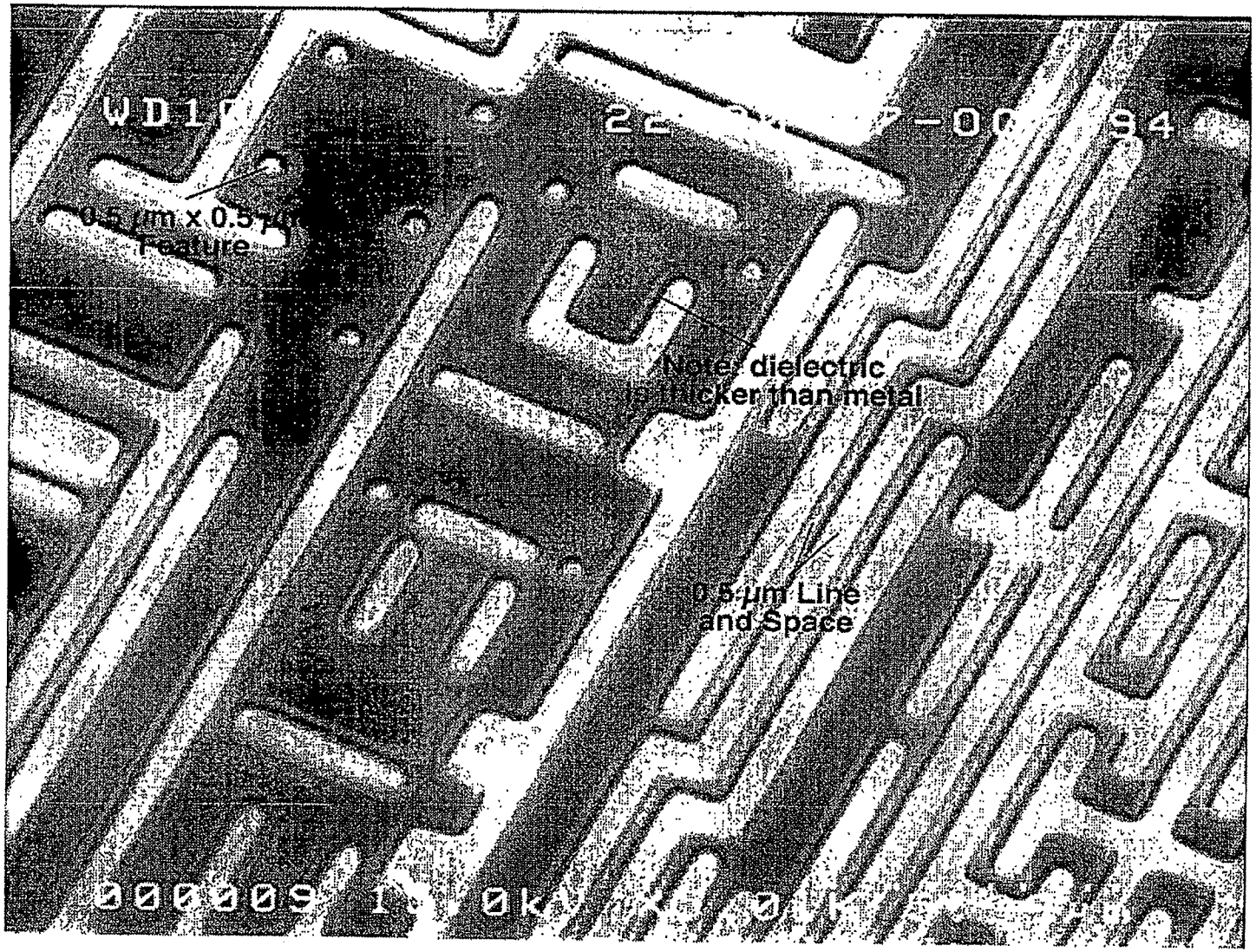
    Tests on the Nb/Au stack are just starting

    Multi-layer metal demonstrations (space transformer) are underway.

Metal Short and OPen Test Structure



Metal Lift-off



Process Status (continued)

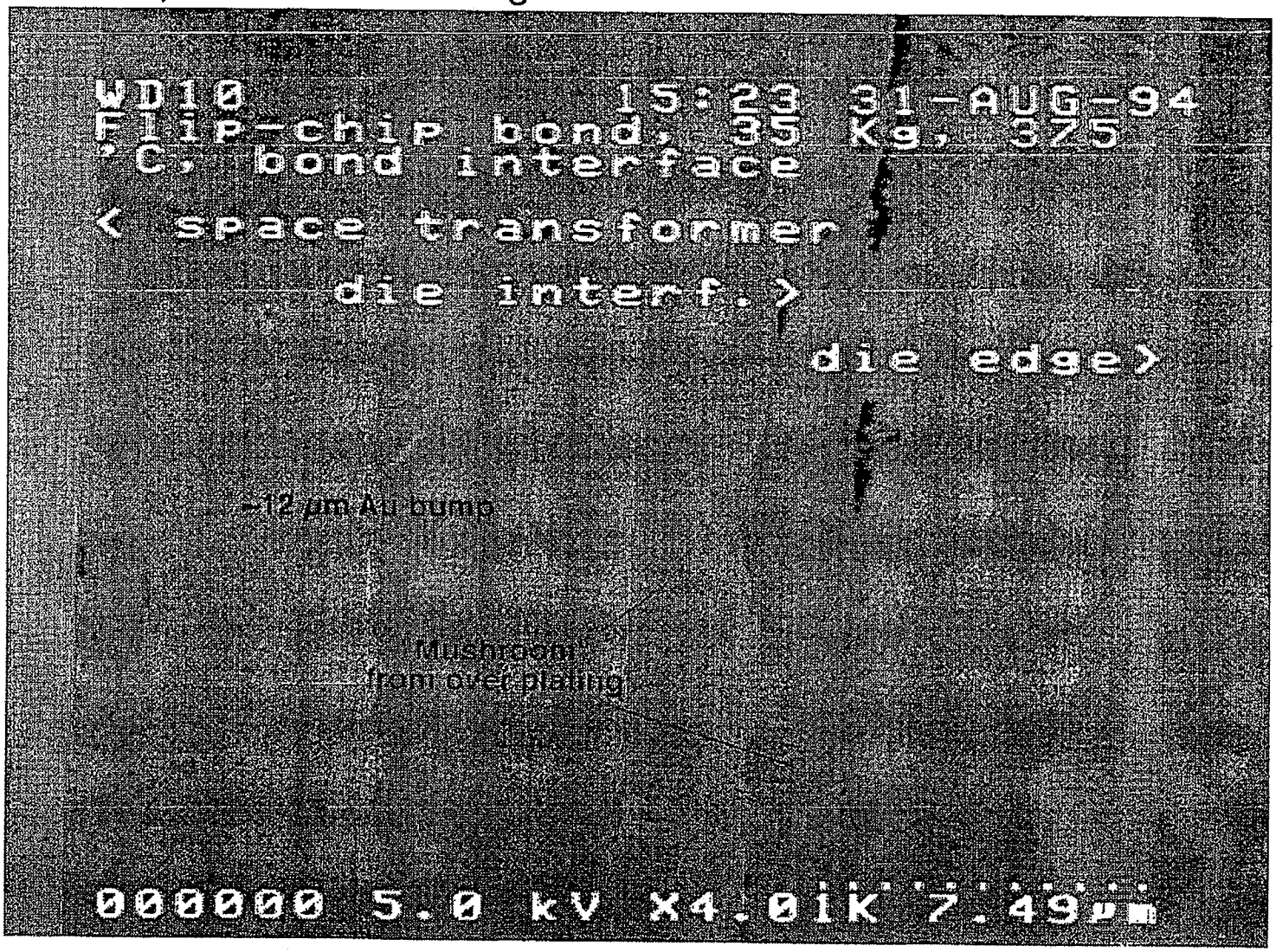
- Packaging
  - — Flip-chip bonding
    - Currently only thermal compression is available - probably not suitable for production devices
    - AuGe eutectic bonding has been tested, equipment modifications are underway and should testing should resume in about a week
    - Pb and PbIn solder methods are being evaluated
    - Production equipment still needs to be specified.
  - — TAB ILB
    - Initial tests are complete
    - System is usable but some issues remain.

Process Status (continued)

- Packaging (continued)
  - — Air bridge

    Initial tests will start next week on forming the air bridge.

Thermocompression Seal Ring



MicroUnity I.C. Process Status

Current Device Status

- Castor/Pollux
  - — Process and circuit test vehicle

    As of 10/28 there are 17 lots of this device in the line with the lead lot at $CoSi_2$ patterning mask.
- Orchis
  - — Yield and burn-in test vehicle, 1 Mbit SRAM

    11 lots in the line, lead lot at SDEC isolation mask.

Current Device Status (continued)

- Calliope
  - — Product I/O device

    11 lots in the line, lead lot at $CoSi_2$ patterning mask.
- Euterpe
  - — Product MPU

    This product is currently in final baseplate verification, we expect reticles for it by mid to late November.

MicroUnity I.C. Process Status

Documentation

- Design Rules
  - — Currently in revision 4.4, 163 pages.
  - — Next revision, 5.0, is due out after initial lots are completed through the line.
- SPICE Model
  - — Current model based on process (SUPREM-4) and device (PISCES-2B) simulations and device characterization from earlier foundry devices.
  - — Models are at the BSIM-2 level.

Documentation (continued)

- Process Specifications
  - All process specifications are on-line in the CIM system
  - The specifications are being written as the process step are stabilized, currently most process steps are running without formal specifications.
- CIM System
  - The system is being written in house, it is a graphically based data base system.
  - Lots are currently tracked and operations verified on the system, lot and equipment comments are being recorded.
  - Video input and equipment status logs are planned.

MicroUnity I.C. Process Status

Summary

- All process equipment is in and running
- Transistors to E-test are expected within about two weeks
- First lots are expected out by the end of November
- First yield should occur within three weeks of the completion of the first lots
- First packaged parts (for physical tests) should be complete by the end of November

Summary (continued)

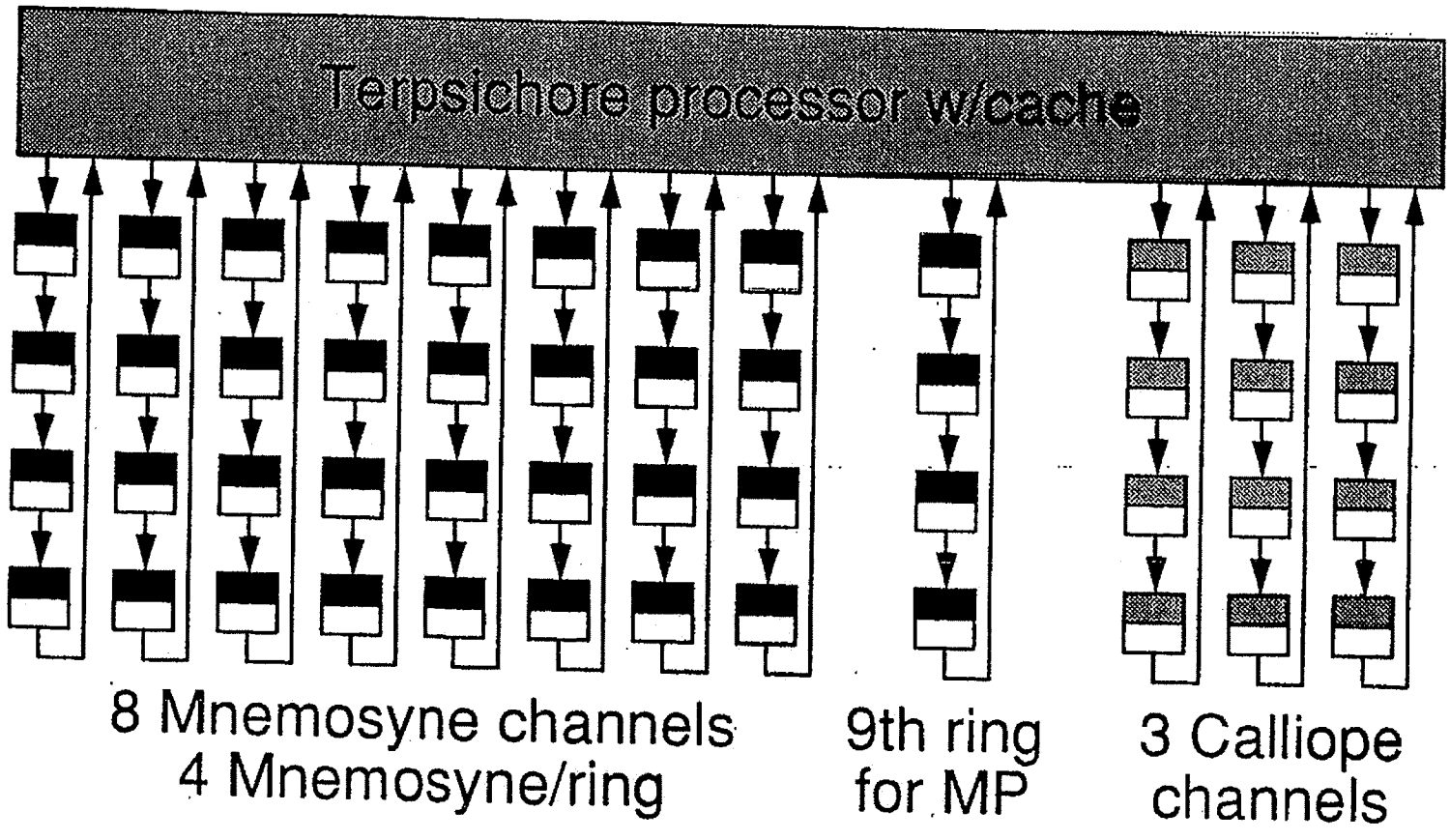
- Process issues to be addressed at this time include:
  - Spacer etch time optimization.
  - SDEC etch back time optimization.
  - Implant adjustment to meet device specifications.
  - Metal lift-off profile control interactions with dielectric stack.
  - Flip-chip bond method evaluation.
  - TAB ILB equipment issues (auto align and TAB finger placement).
  - Air bridge process bring up.

microunity

Characteristics

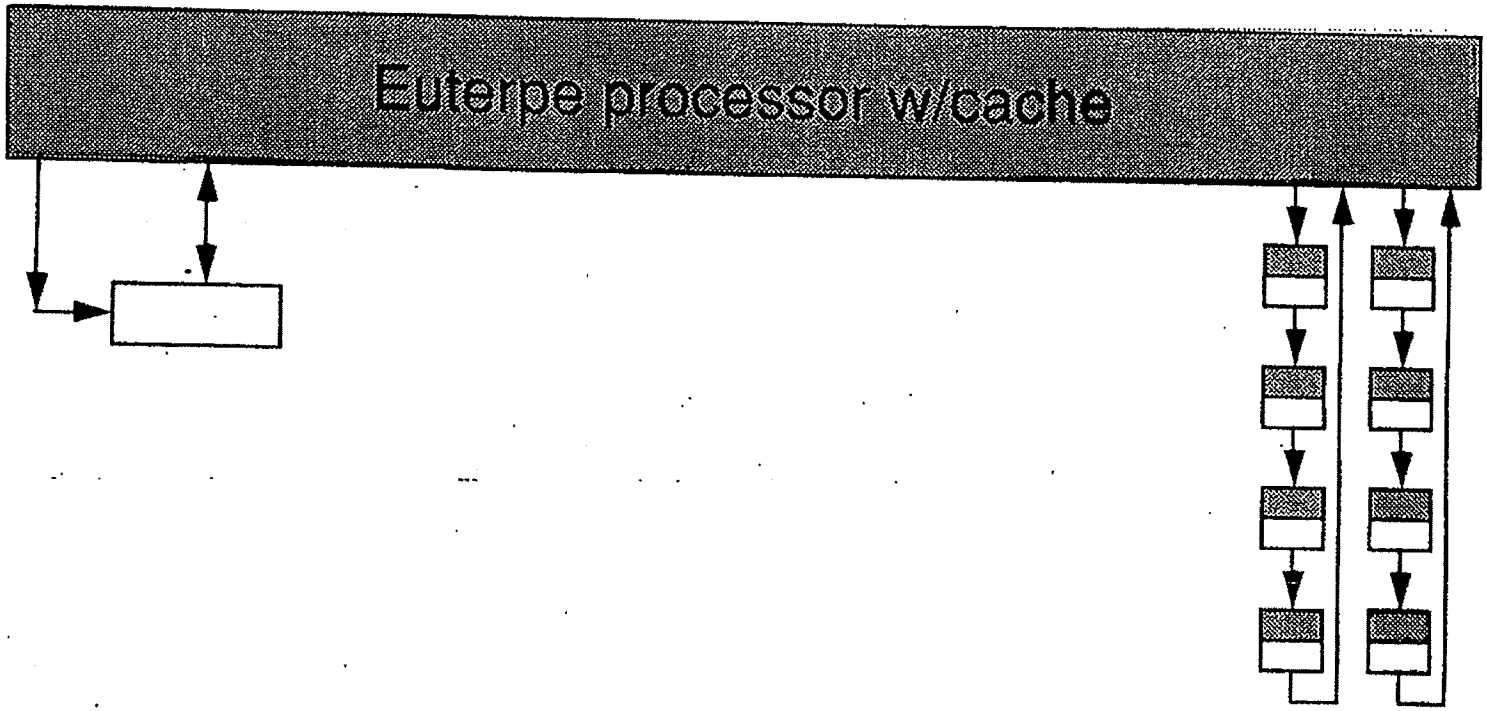
- Byte addressing, 64-bit virtual address space
  - 8-, 16-, 32-, 64-, 128-bit memory transfers
- 64-bit general registers
- 32-bit, aligned instructions
- Simplest possible user state
- High-bandwidth memory
- Precise exceptions

Terpsichore memory structure



microunity

Euterpe memory structure



32-bit SDRAM interface

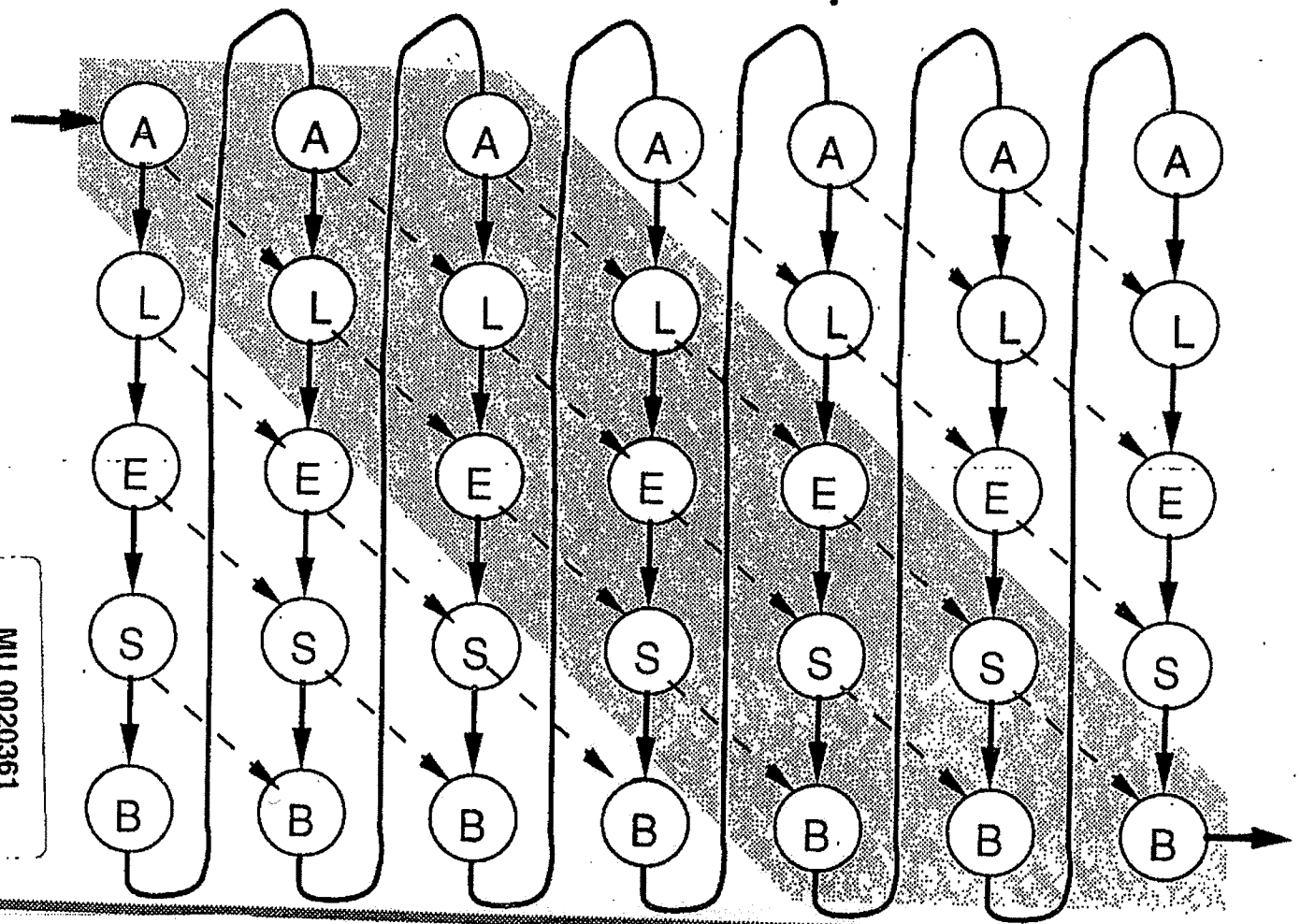
2 Hermes channels

Euterpe subset implementation

- no floating-point
- no interprocessor communications
- no strong/sequential memory ordering
- no unaligned memory access
- 2 Hermes channels, subset of full Hermes interleaving patterns: no octlet and no multiprocessor interleaves
- no EGFMUL64, G{,U}DIV, G{{,U}MUL{,ADD},ADD,SUB,SET}.{2,4}

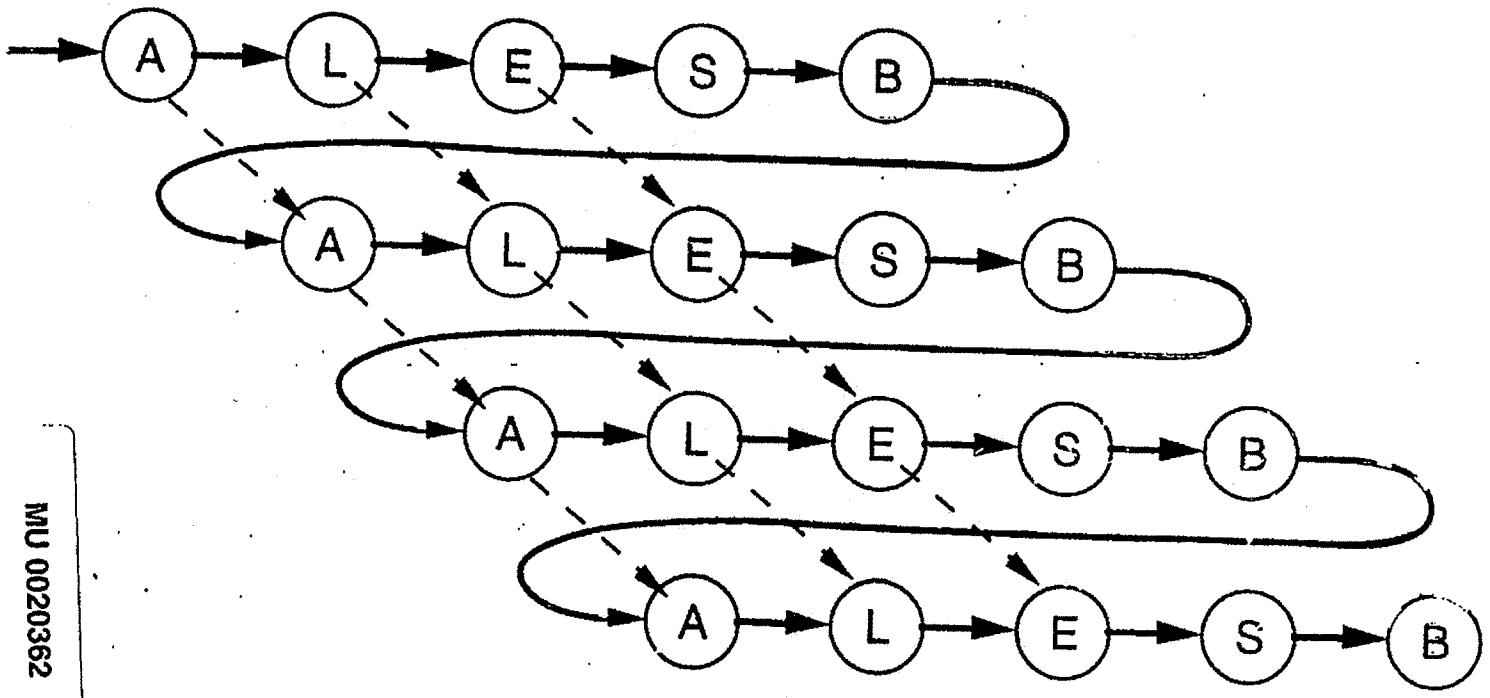
microunity

Superscalar Pipeline

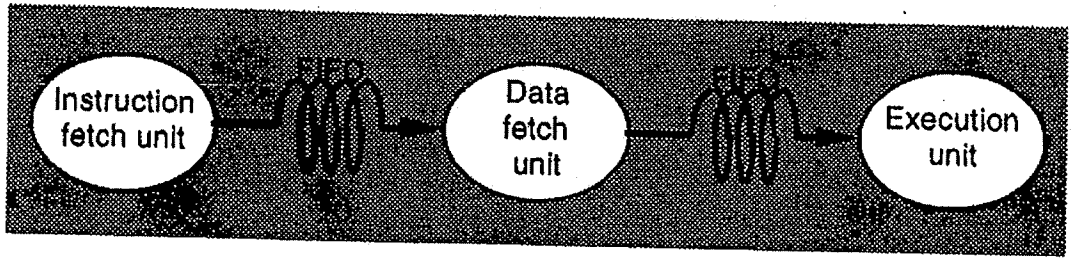


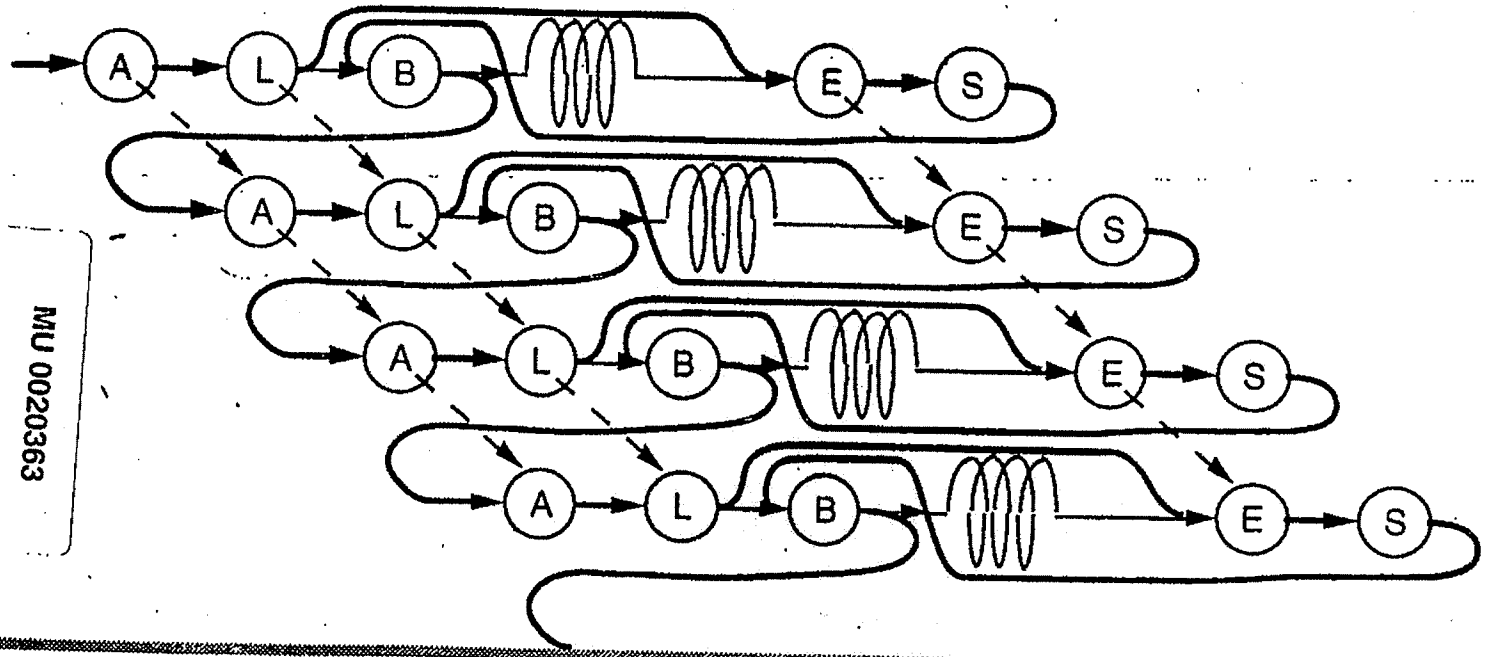
microunity

Superstring Pipeline



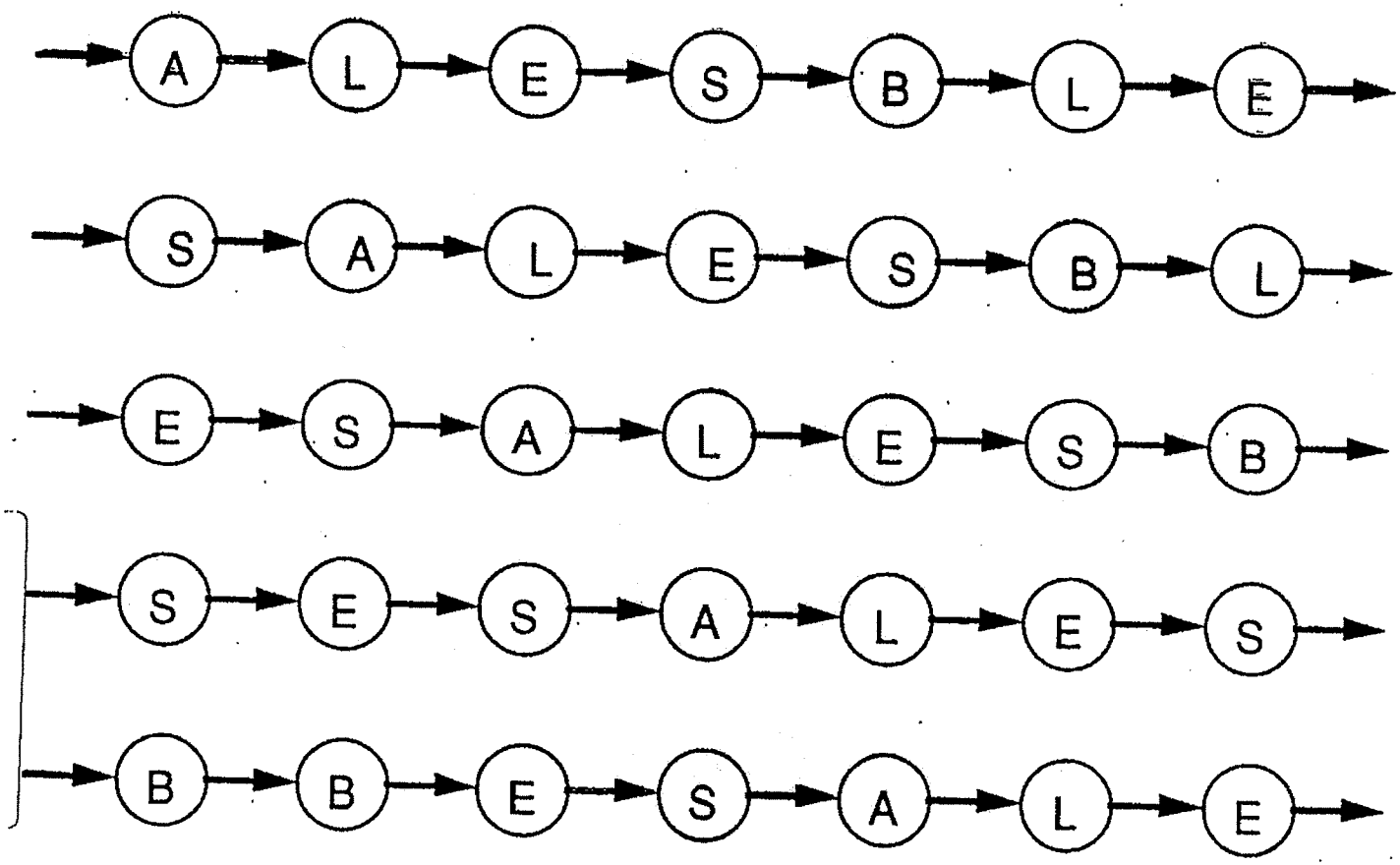
Superspring Pipeline





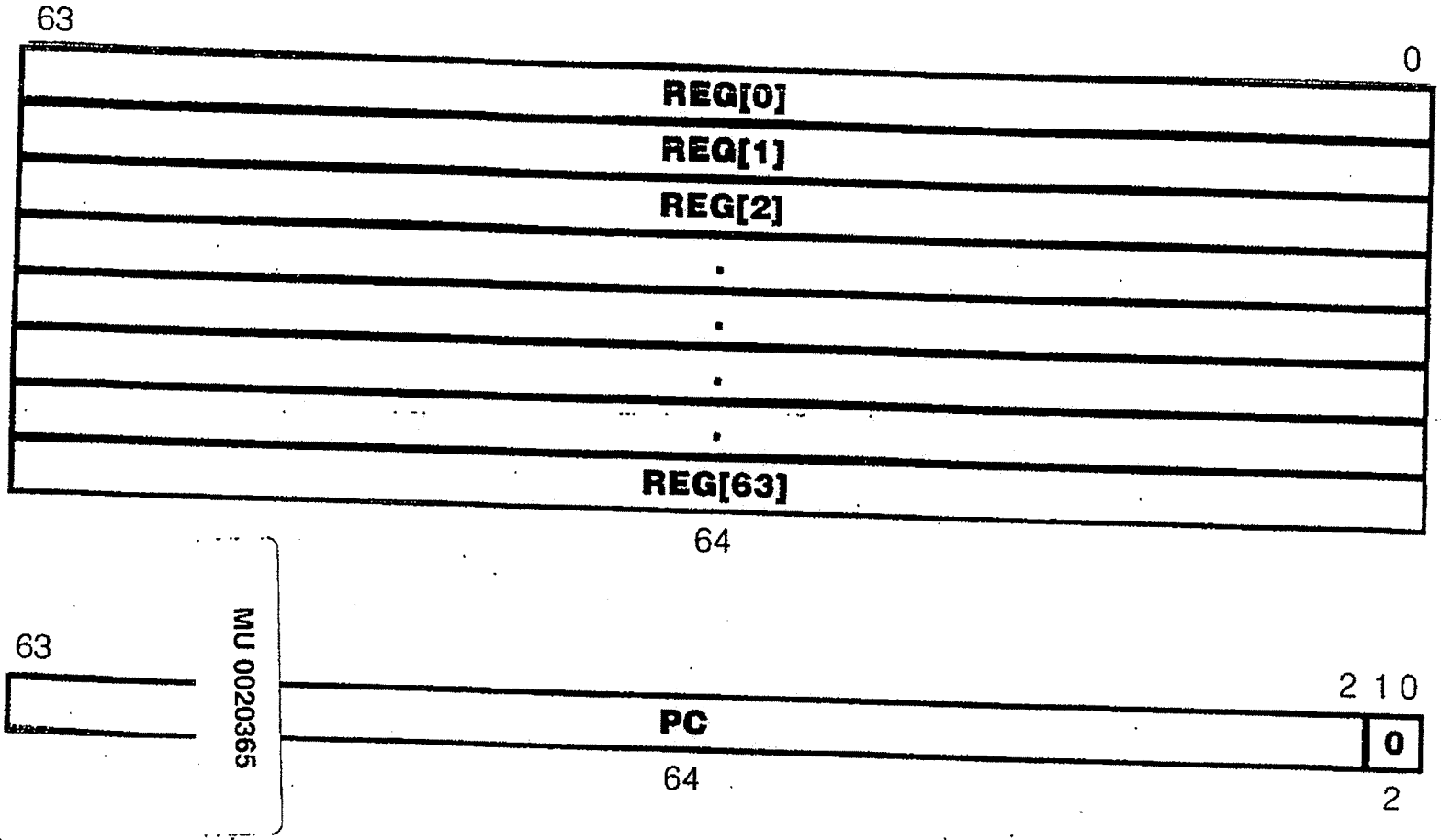
microunity

SuperThread Pipeline



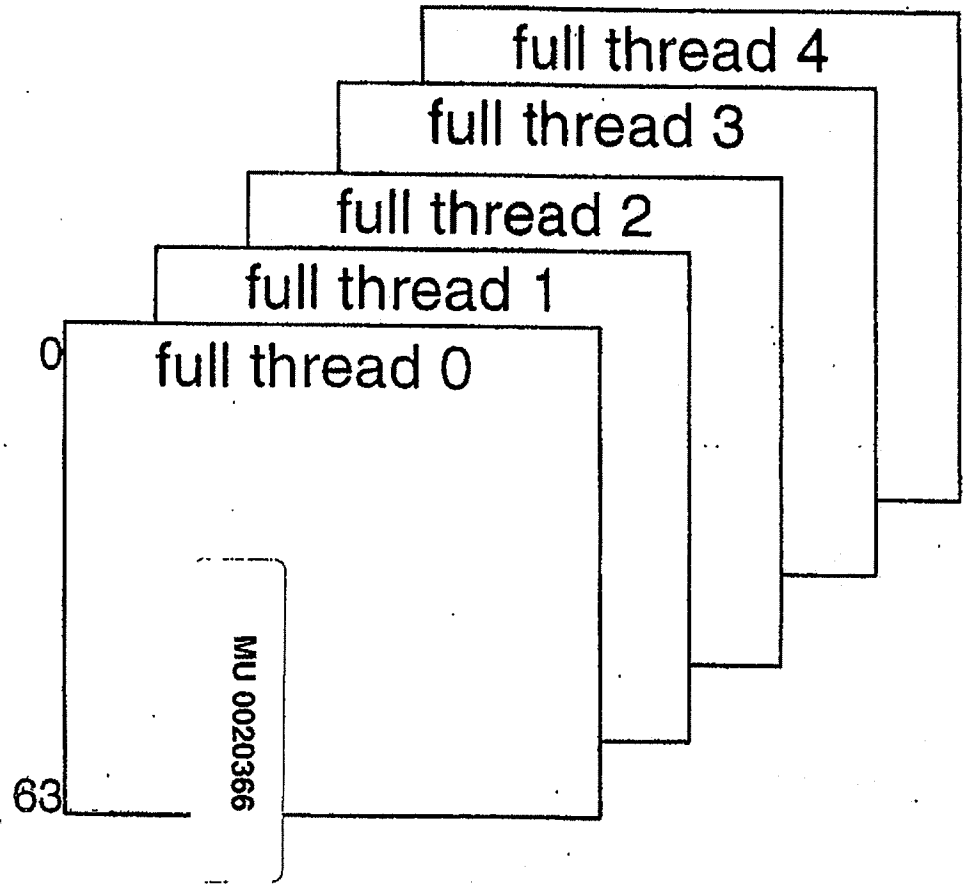
microunity

User state



microunity

SuperThread state



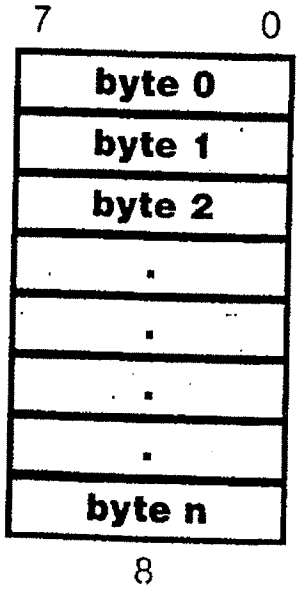
state per thread:

- 64 x 64-bit registers
- 62-bit PC
  - 2-bit privilege level
- 64-bit event mask
- 16-bit LTLB

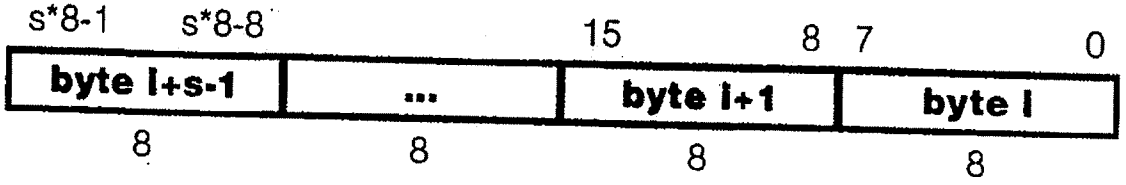
microunity

Data Representation

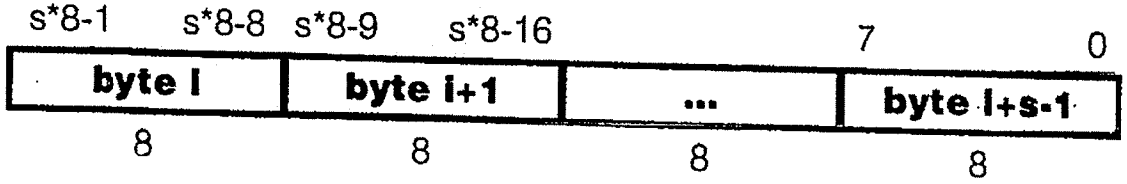
Memory



Little-endian

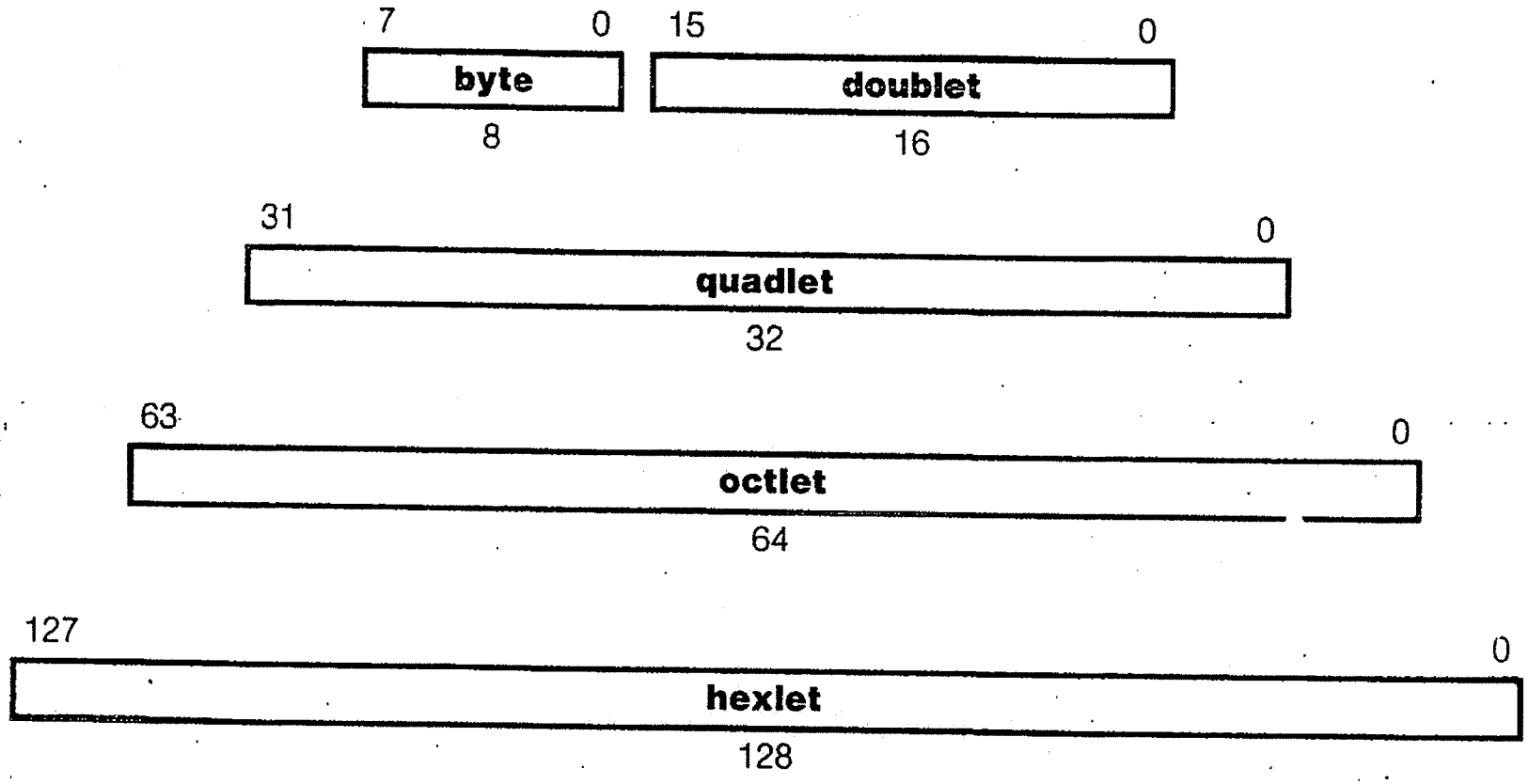


Big-endian



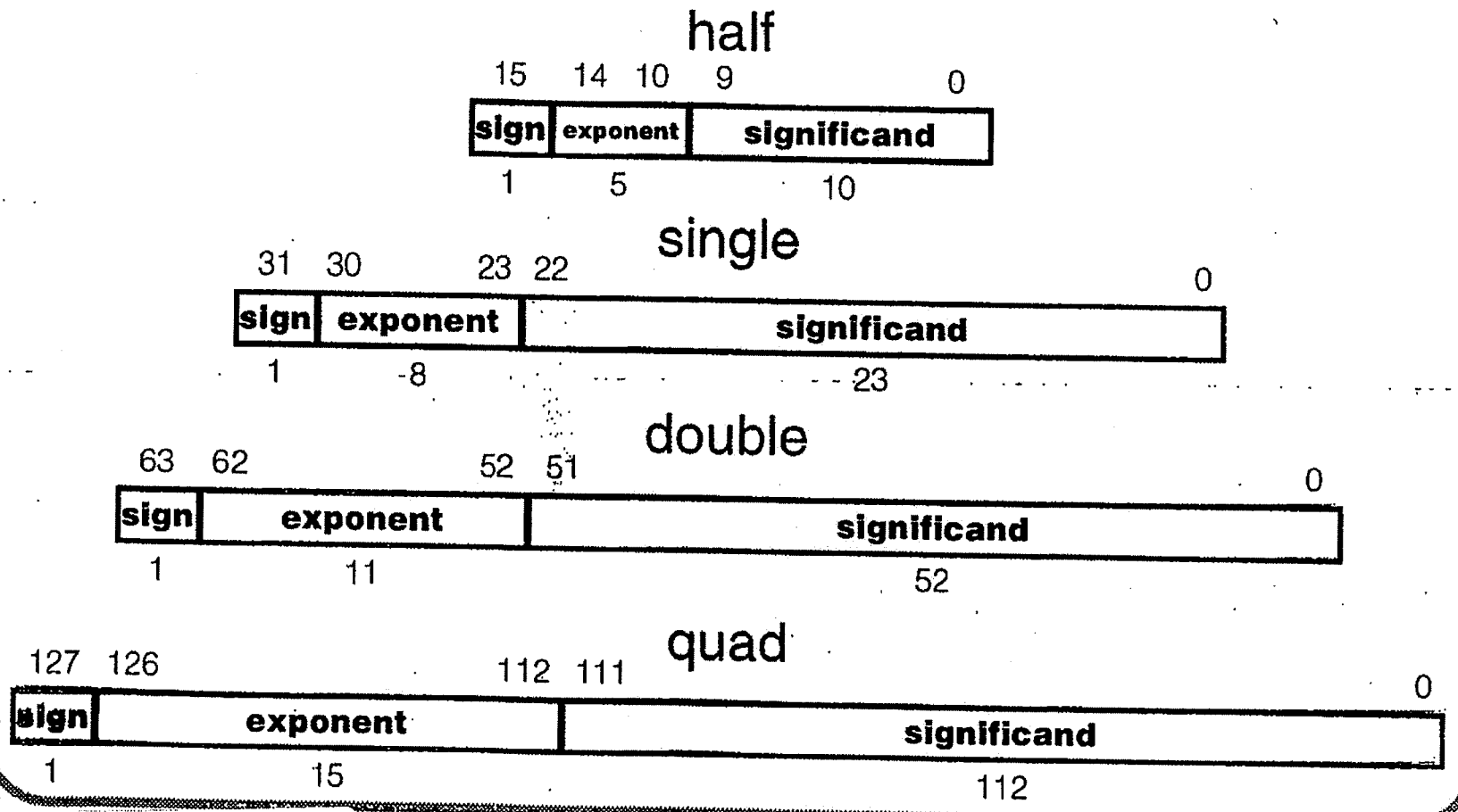
microunity

Fixed-Point Data Sizes

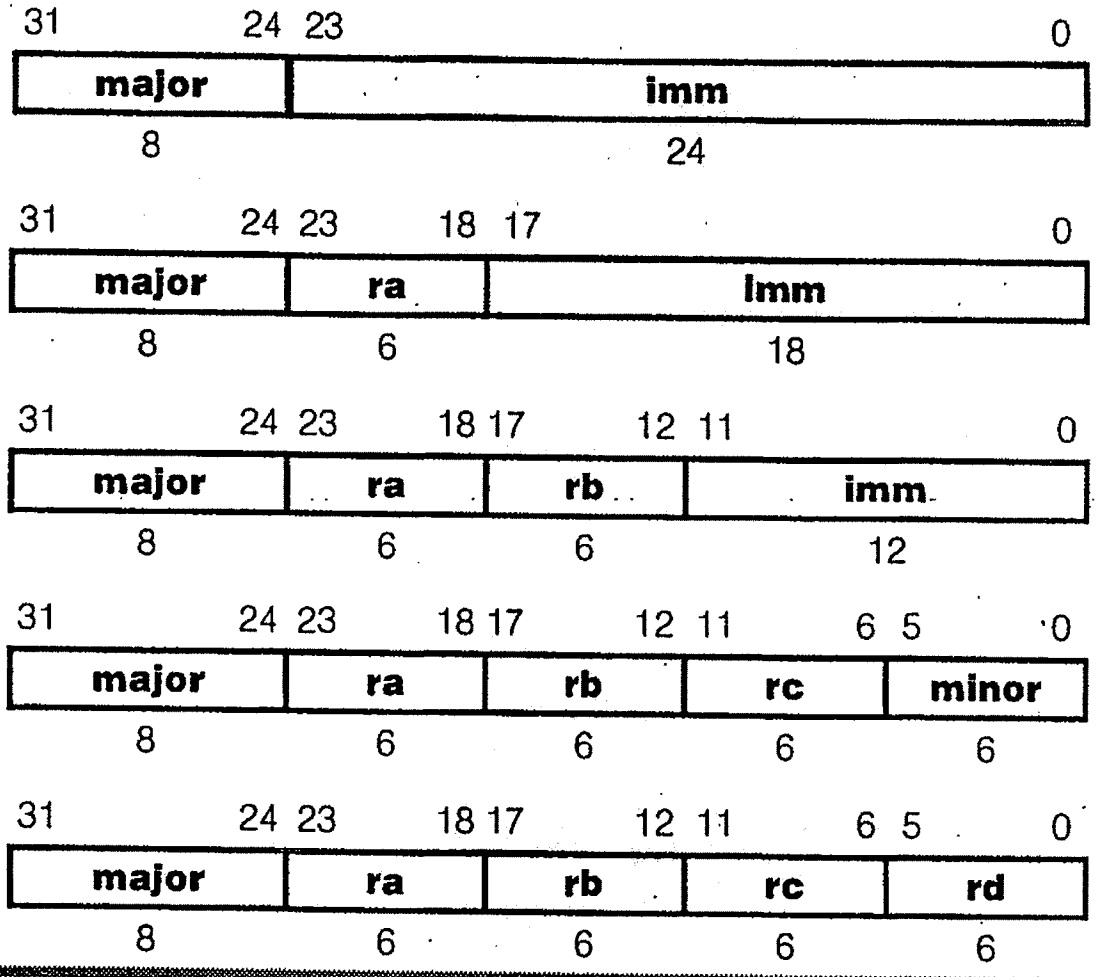


microunity

Floating-point Data Sizes



Instruction Formats



microunity

Major Operation Codes

MAJOR	0	32	64	96	128	160	192	224
0	ERES	GSHUFFLEI	FMULADD16	GMULADD1	LU16LAI	SAAS64LAI	EADDIO	BFE16
1	ESHUFFLEI4MUX	GSHUFFLEI4MUX	FMULADD32	GMULADD2	LU16BAI	SAAS64BAI	EADDIUO	BFNUE16
2		GSELECT8	FMULADD64	GMULADD4	LU16LI	SCAS64LAI	ESETIL	BFNUGE16
3	EMDEPI	GMDEPI		GMULADD8	LU16BI	SCAS64BAI	ESETIGE	BFNUL16
4	EMUX	GMUX	FMULSUB16	GMULADD16	LU32LAI	SMAS64LAI	ESETIE	BFE32
5	E8MUX	G8MUX	FMULSUB32	GMULADD32	LU32BAI	SMAS64BAI	ESETINE	BFNUE32
6	EGFMUL64	GGFMUL8	FMULSUB64	GMULADD64	LU32LI	SMUX64LAI	ESETIUL	BFNUGE32
7	ETRANSPOSE8MUX	GTRANSPOSE8MUX		GEXTRACT128	LU32BI	SMUX64BAI	ESETIUGE	BFNUL32
8					L16LAI	S16LAI	ESUBIO	BFE64
9	ESWIZZLE	GSWIZZLE		GUMULADD2	L16BAI	S16BAI	ESUBIUO	BFNUE64
10		GSWIZZLE.COPY		GUMULADD4	L16LI	S16LI	ESUBIL	BFNUGE64
11		GSWIZZLE.SWAP		GUMULADD8	L16BI	S16BI	ESUBIGE	BFNUL64
12	EDEPI	GDEPI	F.16	GUMULADD16	L32LAI	S32LAI	ESUBIE	BFE128
13	EUDEPI	GUDEPI	F.32	GUMULADD32	L32BAI	S32BAI	ESUBINE	BFNUE128
14	EWTHI	GWTHI	F.64	GUMULADD64	L32LI	S32LI	ESUBIUL	BFNUGE128
15	EUWTHI	GUWTHI		GUEXTRACT128	L32BI	S32BI	ESUBIUGE	BFNUL128
16			GFMULADD16	GEXTRACTI	L64LAI	S64LAI	EADDI	BANDE
17			GFMULADD32	GEXTRACTI16	L64BAI	S64BAI	EXORI	BANDNE
18			GFMULADD64	GEXTRACTI32	L64LI	S64LI	EORI	BL/BLZ
19			GFMULADD128	GUEXTRACTI64	L64BI	S64BI	EANDI	BGE/BGEZ
20			GFMULSUB16	GEXTRACT	L128LAI	S128LAI	ESUBI	BE
21			GFMULSUB32	.I.64	L128BAI	S128BAI		BNE
22			GFMULSUB64	G.EXTRACT	L128LI	S128LI	ENORI	BUL/BGZ
23			GFMULSUB128	.I.128	L128BI	S128BI	ENANDI	BUGE/BLEZ
24				G.1	L8I	S8I		BGATEI
25				G.2	LU8I			
26				G.4				
27				G.8				
28			GF.16	G.16			ECOPYI	BI
29			GF.32	G.32				BLINKI
30			GF.64	G.64				
31			GF.128	G.128	L.MINOR	S.MINOR	E.MINOR	B.MINOR

CONFIDENTIAL

MU 0020371

CCH

REDACTED

Minor Operation Codes: F, GF

F.size	0	8	16	24	32	40	48	56
0	FADD.N	FADD.T	FADD.F	FADD.C	FADD	FADD.X	FSETE	FSETE.X
1	FSUB.N	FSUB.T	FSUB.F	FSUB.C	FSUB	FSUB.X	FSETNUE	FSETNUE.X
2	FMUL.N	FMUL.T	FMUL.F	FMUL.C	FMUL	FMUL.X	FSETNUGE	FSETNGE.X
3	FDIV.N	FDIV.T	FDIV.F	FDIV.C	FDIV	FDIV.X	FSETNUL	FSETNUL.X
4	F.UNARY.N	F.UNARY.T	F.UNARY.F	F.UNARY.C	F.UNARY	F.UNARY.X		
5								
6								
7								

GF.size	0	8	16	24	32	40	48	56
0	GFADD.N	GFADD.T	GFADD.F	GFADD.C	GFADD	GFADD.X	GFSETE	GFSETE.X
1	GFSUB.N	GFSUB.T	GFSUB.F	GFSUB.C	GFSUB	GFSUB.X	GFSETNUE	GFSETNUE.X
2	GFMUL.N	GFMUL.T	GFMUL.F	GFMUL.C	GFMUL	GFMUL.X	GFSETNUGE	GFSETNGE.X
3	GFDIV.N	GFDIV.T	GFDIV.F	GFDIV.C	GFDIV	GFDIV.X	GFSETNUL	GFSETNUL.X
4	GF.UNARY.N	GF.UNARY.T	GF.UNARY.F	GF.UNARY.C	GF.UNARY	GF.UNARY.X		
5								
6								
7								

F.UNARY.size.r	
0	F.ABS
1	F.NEG
2	F.SQR
3	
4	F.SINK
5	F.FLOAT
6	F.INFLATE
7	F.DEFLATE

GF.UNARY.size.r	
0	GF.ABS
1	GF.NEG
2	GF.SQR
3	
4	GF.SINK
5	GF.FLOAT
6	GF.INFLATE
7	GF.DEFLATE

microunity

Minor Operation Codes: E, G

E.MINOR	0	8	16	24	32	40	48	56
0	EADDO	ESUBO	EANDN		EADD	ESUB	ESHLIO	ESHRI
1	EADDUO	ESUBUO	EXOR		ESHLO	ESHLUO		
2	ESETL	ESUBL	EOR				ESHLIUO	EUSHRI
3	ESETGE	ESUBGE	EAND		ELMS	EULMS		
4	ESETE	ESUBE	EORN		EASUM	ESELECT8	ESHUFFLEI	EROTRI
5	ESETNE	ESUBNE	EXNOR		EROTL	ESHL		
6	ESETUL	ESUBUL	ENOR		ESHR	EUSHR	ESHLI	EMSHRI
7	ESETUGE	ESUBUGE	ENAND		EROTR	EMSHR		

G.size	0	8	16	24	32	40	48	56
0		GMUL	GANDN		GADD	GSUB	GEXPAND	GSHR
1		GUMUL	GXOR		GCOMPRESS	GUCOMPRESS	.I	.I
2	GSETL	GDIV	GOR				GUEXAPAND	GUSHR
3	GSETGE	GUDIV	GAND				.I	.I
4	GSETE	GSUB	GORN		GEXPAND	GUEXPAND	GCOMPRESSI	GROTR
5	GSETNE		GXNOR		GROTL	GSHL	GUCOMPRESS	.I
6	GSETUL		GNOR		GSHR	GUSHR	GSHL	GMSHR
7	GSETUGE		GNAND		GROTR	GMSHR	.I	.I

CONFIDENTIAL

MU 0020373

CCH REDACTED

Minor Operation Codes: L, S, B

L.min	0	8	16	24	32	40	48	56
0	LU16LA	L16LA	L64LA	L8				
1	LU16BA	L16BA	L64BA	LU8				
2	LU16L	L16L	L64L					
3	LU16B	L16B	L64B					
4	LU32LA	L32LA	L128LA					
5	LU32BA	L32BA	L128BA					
6	LU32L	L32L	L128L					
7	LU32B	L32B	L128B					

S.min	0	8	16	24	32	40	48	56
0	SAAS64LA	S16LA	S64LA	S8				
1	SAAS64BA	S16BA	S64BA					
2	SCAS64LA	S16L	S64L					
3	SCAS64BA	S16B	S64B					
4	SMAS64LA	S32LA	S128LA					
5	SMAS64BA	S32BA	S128BA					
6	SMUX64LA	S32L	S128L					
7	SMUX64BA	S32B	S128B					

B.MINOR	0	8	16	24	32	40	48	56
0	B							
1	B.LINK							
2	B.DOWN							
3								
4			BGATE	BBACK				
5								
6								
7								

microunity

Branches

- Non-delayed branches
- Fixed-point compare and branch
  - equal, not equal, less, or greater/equal
  - two-operand signed or unsigned compare
  - bitwise and, then compare vs. zero
- Floating-point compare and branch
  - Classic comparisons
  - IEEE-aware comparisons
  - half, single, double, or quad precision
- Unconditional branch
  - pc+offset or register
  - save link (register 0)

Floating-point Compare

Mnemonic		Branch taken if values compare as:				Exception if unordered
code	C	Unordered	Greater	Less	Equal	
E	==	F	F	F	T	no
NUGE	!?>=	F	F	T	F	no
NUL	!?<	F	T	F	T	no
UL	?<	T	F	T	F	no
UGE	?>=	T	T	F	T	no
NE	!=	T	T	T	F	no
L,NGE	<,!>=		F	T	F	yes
GE,NL	<=,!>		T	F	T	yes

Privilege-level crossing branches

- Four privilege levels held in least-significant peck of PC
- Branch gateway

  secure equivalent to L128LI+B+increase in privilege
- Branch down

  secure equivalent to B+decrease in privilege
- Branch back

  secure equivalent to L128LI+B+decrease in privilege

  permits complete restoration of register state after event

Loads, Stores

- Byte addressing
- Big-endian or little-endian
- Byte, doublet, quadlet, octlet, hexlet
- Signed or unsigned (byte, doublet, quadlet)
- Aligned or unaligned (doublet, quadlet, octlet, hexlet)
- Base register + 12-bit signed offset

  Base register + index register
- Large immediates are loaded, not constructed

Synchronization

- Sequentially consistent and weak ordering
  - Specified in TLB entry
  - Synchronization operations always sequentially consistent
- Aligned octlet operations
- Swap (load mem->reg, store reg->mem)
- Add (load mem->reg, add reg+mem->mem)
- Compare&Swap (load mem->reg, compare reg<->reg, if equal, store reg->mem)
- Masked-write (load mem->reg, mux:mask,reg,mem->mem)

Fixed-point

- Shifts, add, subtracts
- Explicit overflow checking
- Bitwise logical operations
- Compare and set boolean
- Register or 12-bit signed immediate
- Integer multiply and divide

microunity

Floating-point

- Half, single, double, quad precision
- Add, sub, mul, div, sqr, abs, neg
- Combined multiply, add/subtract
- Format conversions
- Explicit rounding selection
- Explicit exception handling
- Explicit inexact checking

Special-Purpose Instructions

- Find most significant one
- Count ones
- Bitwise multiplex
- Deal & Shuffle
- Gather & Scatter
- Galois Field Multiply

Find most/least significant one

E.ULMS rc,ra

```
t ← REG[ra]
if t = 0
      res ← -1
else
      res ← i :: (t   =1 and t          =0)
endif              i          63..i+1
REG[rc] ← res
```

Most-significant:

E.ULMS rt,rs

Least-significant:

```
E.ADDI   rt,rs,-1
E.ANDN   rt,rt,rs
E.ULMS   rt,rt
```

Count Ones

E.ASUM rc,ra,rb

```
t ← REG[ra] & REG[rb]
res ← 0
for i ← 0..63
      res ← res + t
endfor
REG[rc] ← res
```

Count Ones:

E.ASUM rt,rs,rs

Multiplex

E.MUX rd,ra,rb,rc

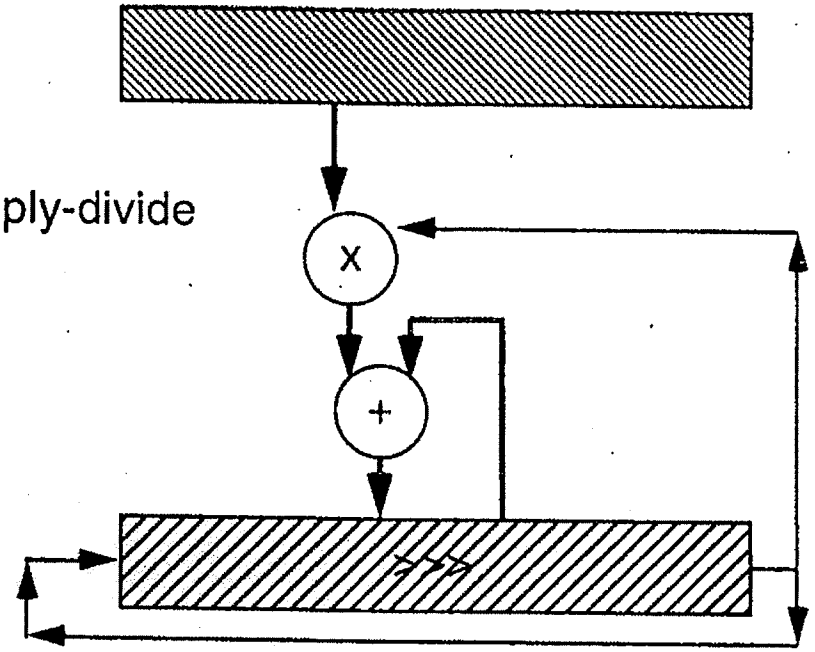
```
t ← REG[ra]
REG[rd] ← (t & REG[rb]) | (~t & REG[rc])
```

G.MUX rd,ra,rb,rc

```
t ← REG[ra]||REG[ra+1]
REG[rd]||REG[rd+1] ← (t & (REG[rb] || REG[rb+1]) ) |
                     (~t & (REG[rc] || REG[rc+1]) )
```

Galois Field Arithmetic

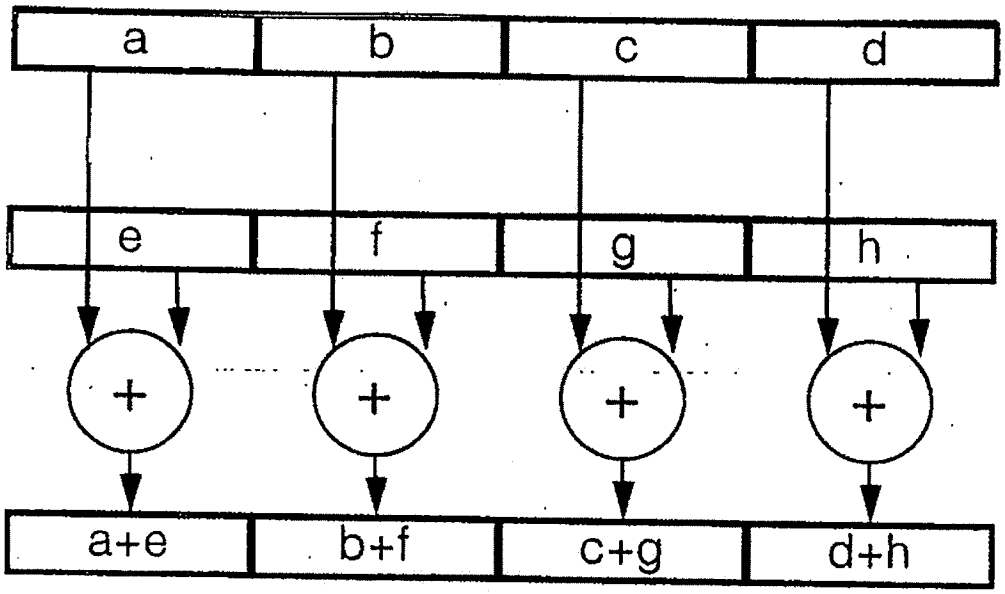
- E.GFMUL.64
  - $GF(2^{64})$ multiply
  - 64-bit polynomial multiply-divide
- G.GFMUL.8
  - $GF(2^{8})$ multiply
  - 8-bit polynomial multiply-divide



Group (DSP) Operations

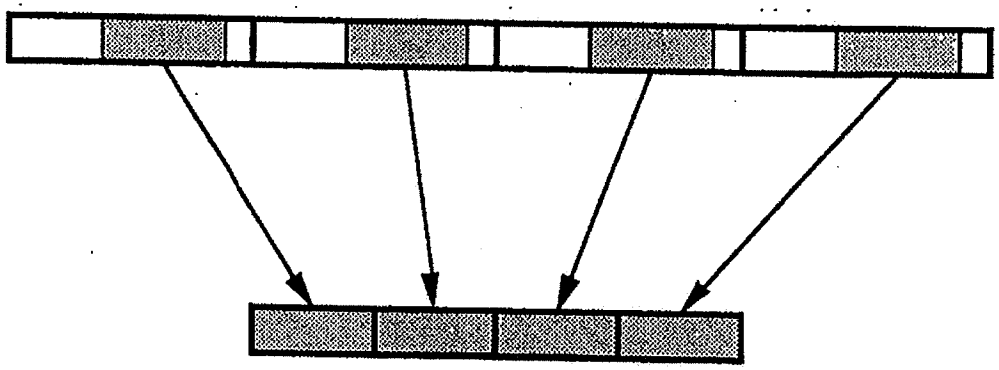
- Designed to be accessible to compilers
- Operate on 128 bit vectors
- Fixed-point data sizes 1, 2, 4, 8, 16, 32, 64 bits
- Floating-point data sizes 16, 32, 64 bits
- Multiply, add/subtract, shift/rotate
- Combined multiply, add/subtract
- Flexible size and format conversion

Group Add



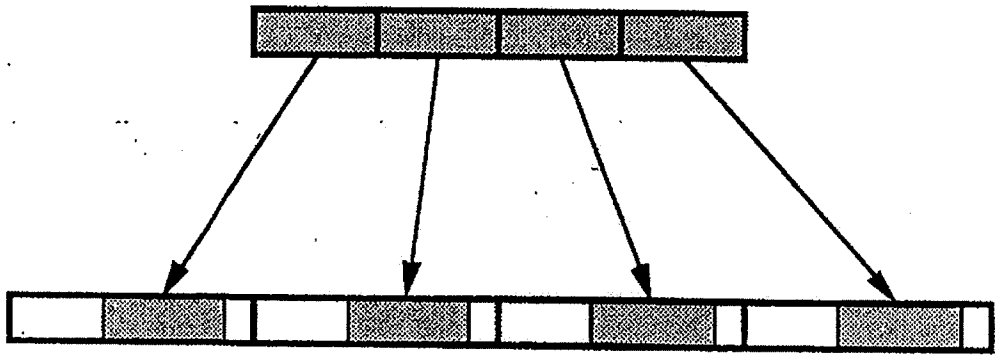
Group Compress, Extract

- Group Compress: 128 bits to 64 bits
  - immediate and dynamic shift amounts for all sizes: 1-64 bits
- Group Extract: 256 bits to 128 bits
  - immediate shift amounts for all sizes: 1-128 bits
  - dynamic shift amounts for 128 bits



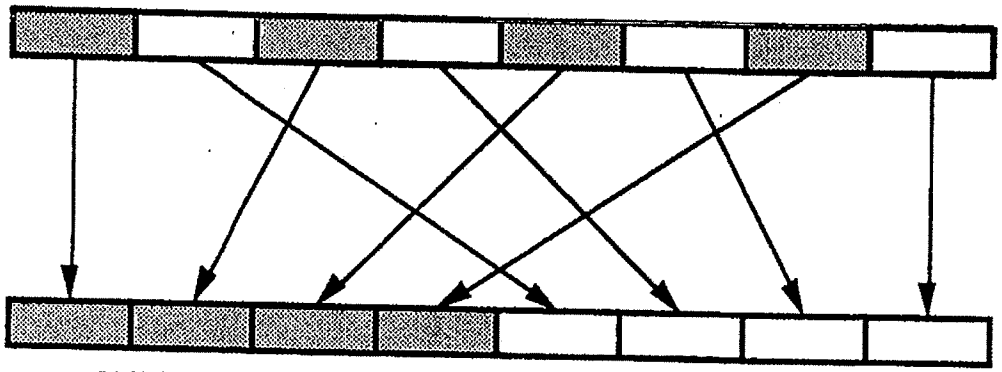
Group Expand

- Group Expand: 64 bits to 128 bits
  - immediate and dynamic shift amounts for all sizes: 1-64 bits
  - signed and unsigned expand

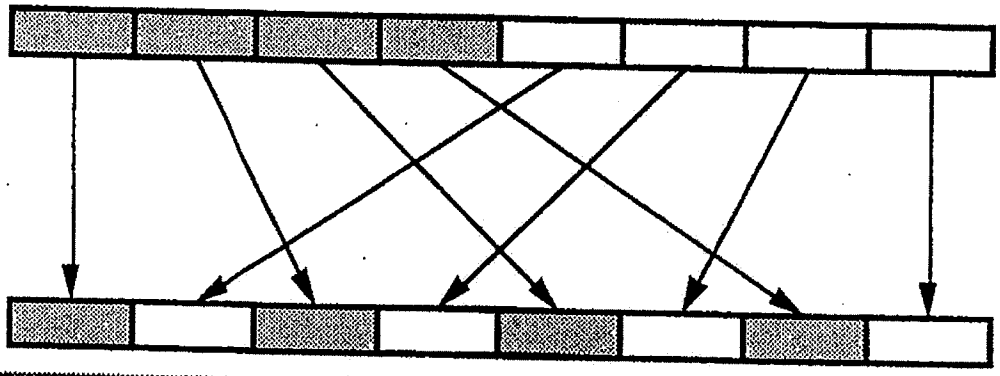


Group Deal, Shuffle

- Group Deal: 128 bits to 128 bits

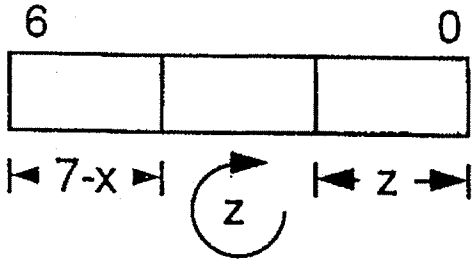


- Group Shuffle: 128 bits to 128 bits



Group Shuffle

- General form: GSHUFFLEI.2^x.2^y.2^z



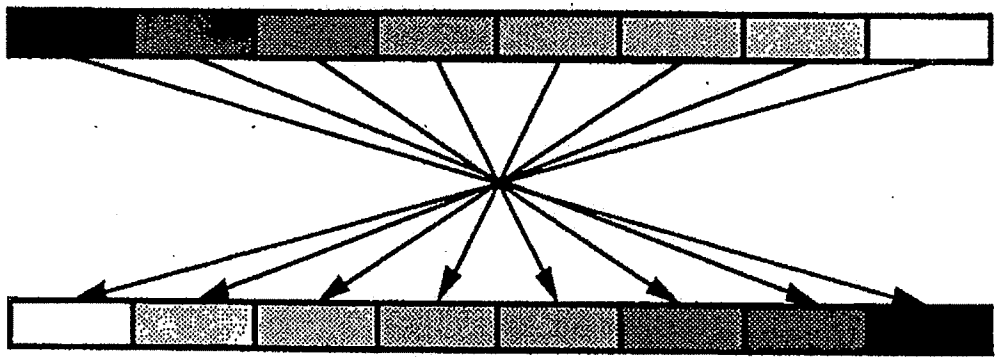
$$imm = (x^3-3x^2-4x)/6 - (z^2-z)/2 + xz + y + 1$$

Group Shift

- Group Shift: 128 bits
  - shift or rotate at 2, 4, 8, 16, 32, 64, 128 bit granularity
  - dynamic: ROTL, ROTR, SHL, SHR, USHR, MSHR
  - immediate: ROTRI, SHLI, SHRI, USHRI, MSHRI
- Group Deposit/Withdraw: 128 bits
  - deposit or withdraw at 2, 4, 8, 16, 32, 64, 128 bit granularity
  - field_size from 1..size, shift_amount from 0..field_size
  - immediate field_size and shift_amount only
  - DEPI, UDEPI, WTHI, UWTHI, MDEPI

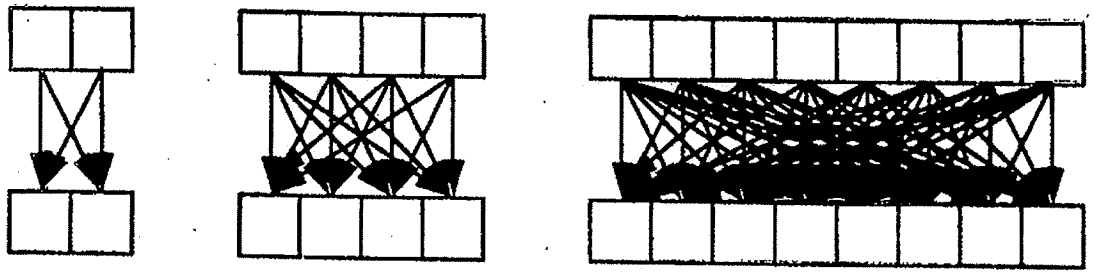
Group Swizzle (Copy-Swap)

- Group Swizzle (Copy-Swap): 128 bits
  - copy and/or swap at 1, 2, 4, 8, 16, 32, 64 bit granularity



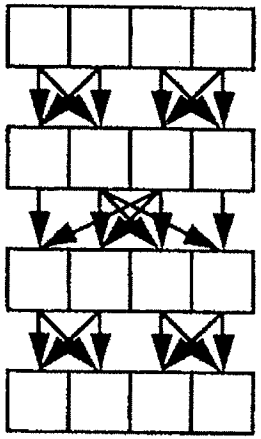
Group Permute

2-mux 4-mux 8-mux

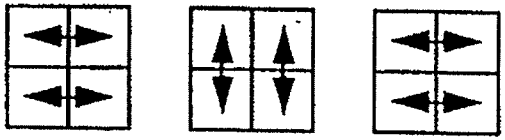


Benes network: multistage permutation

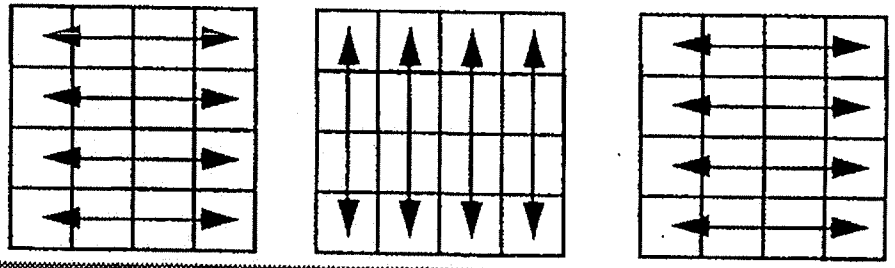
4-mux, 3 stage



4-mux, two-dimensional (3-stage)



16-mux, two-dimensional (3-stage)



Group Permute

- two dimensional network

  8-mux, 16-mux, 8-mux

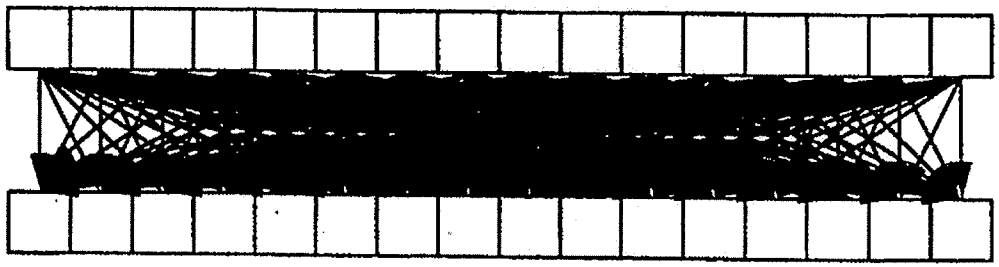
  same network used for shifts, rotates, shuffles, permute

  network itself capable of arbitrary permute, but instructions can't provide sufficient control bits in a single instruction

- G.SELECT.8

  128 bits data, 4x16=64 bits control

  16-way mux, byte-level granularity: complete byte permute 16-mux



microunity

Group Permute

- G.SHUFFLEI.4MUX
  - 128 bits data, 2x64 bits control
  - 4-way mux with shuffle
  - 3 passes perform complete 16-bit permute
  - 5 passes perform complete 64-bit permute
- G.8MUX, G.TRANSPOSE.8MUX
  - 128 bits data, 3x64 bits control
  - 8-way mux with optional transpose (triple shuffle)
  - 3 passes perform complete 64-bit permute

System Facilities

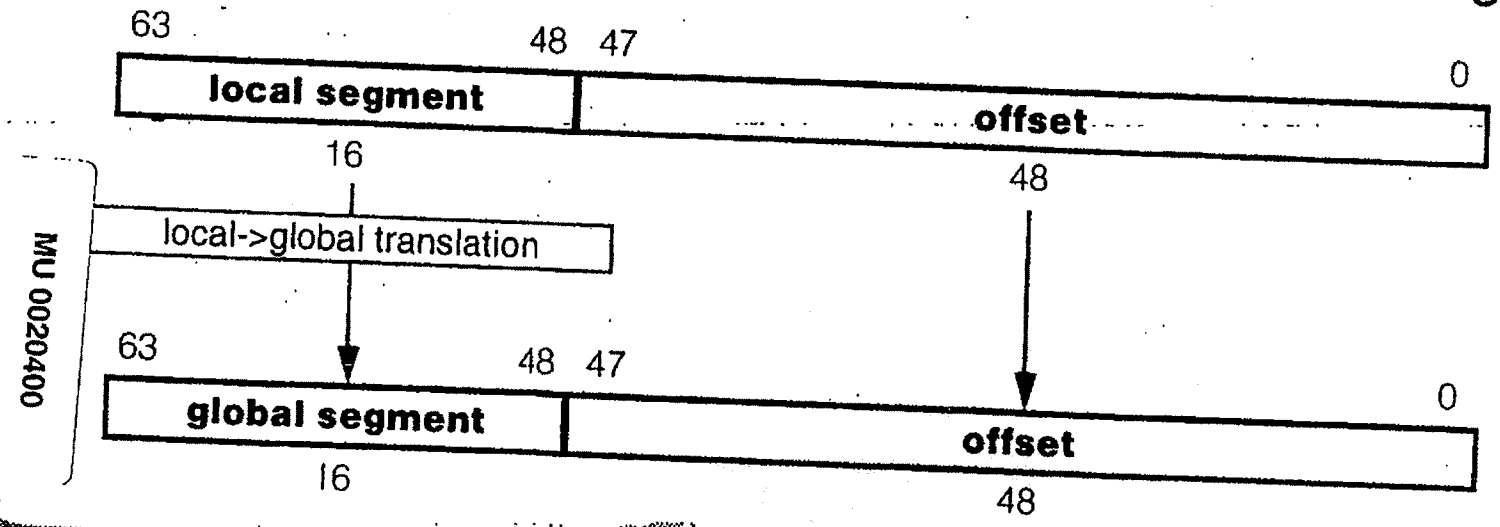
- All system state memory-mapped
- All system code can be compiled
- Lightweight exception and event handling
- Protected gateways
- Virtual-addressed, virtual/physical-tagged internal caches
- Internal buffer memory
  - Cache tags
  - Interprocessor communication buffers
  - I/O transfer buffers

Virtual Memory

- Arbitrary virtual to physical maps
  - any page size
  - frame buffer, physical kernel spaces use one TLB entry each
  - allocation of physically interleaved memory to virtual space
- 64-bit virtual addresses
- Virtual caches with support for aliases
- up to 4 privilege levels, in TLB
- up to 16 bit address space identifiers
- asid part of virtual address

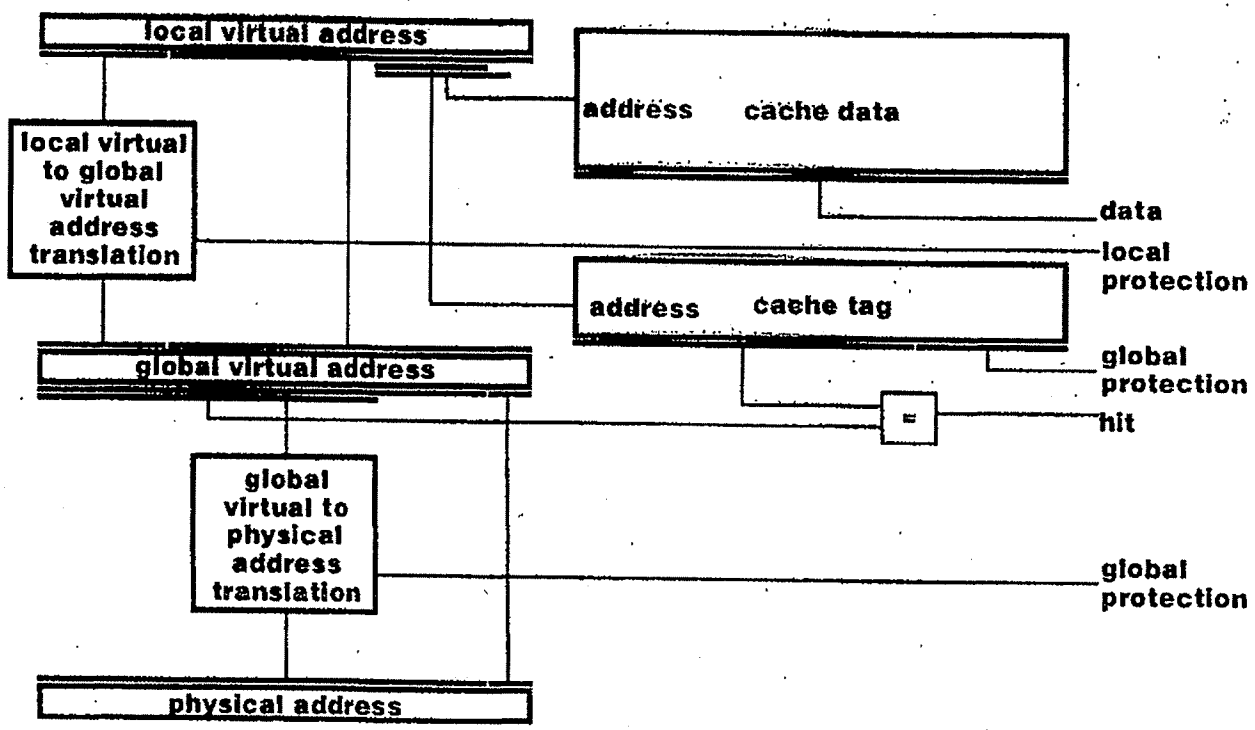
Need VM space be > 64 bits?

- 64 bit space is more than large enough
- Segmentation vs matching
- UNIX fork requires process-local addressing
- kernel and library code prefers global addressing



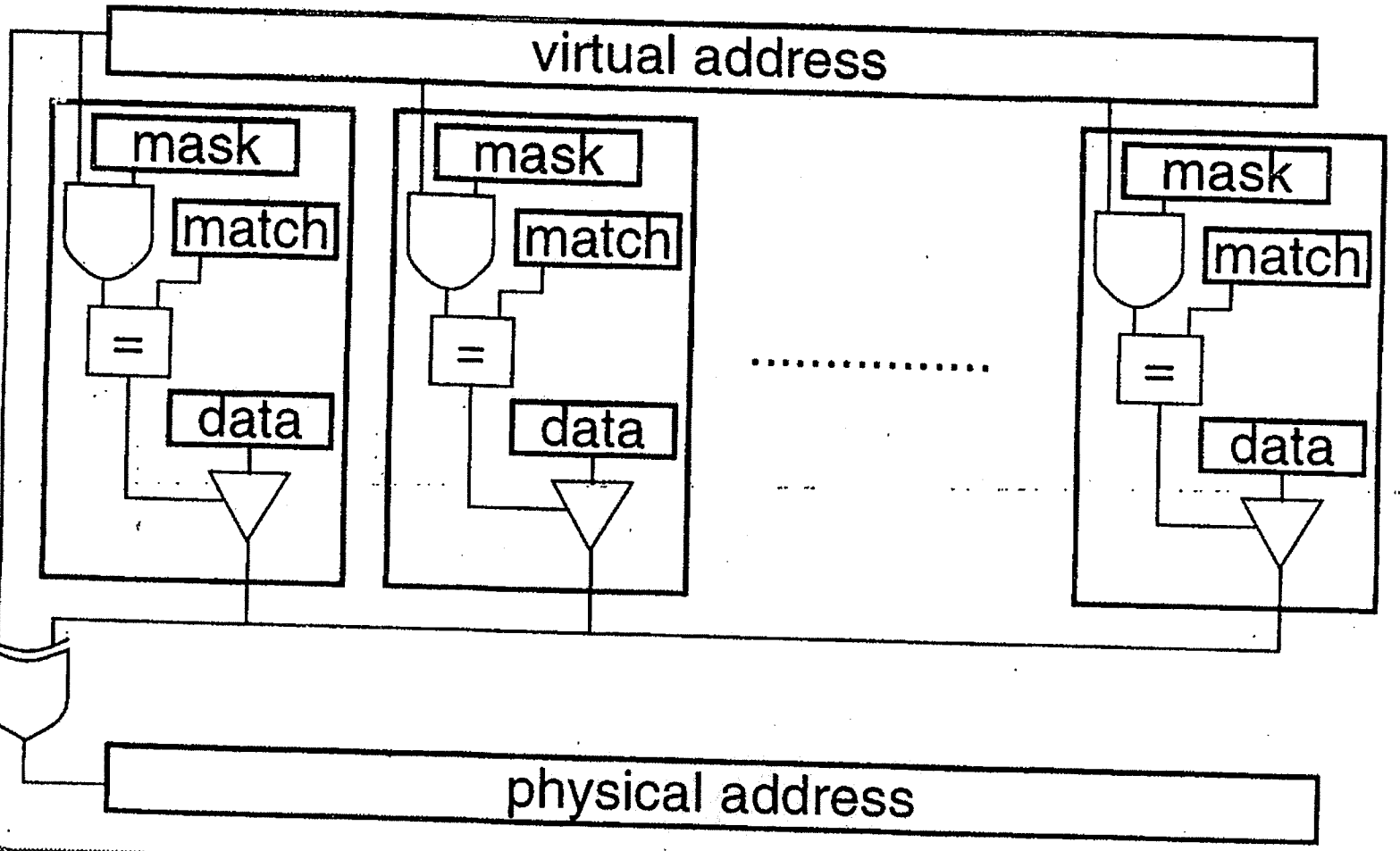
microunity

Translation Block Diagram

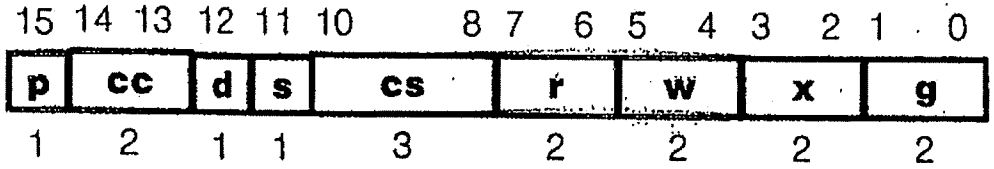


microunity

Translation Lookaside Buffer



Protection information



- r,w,x,g: minimum privilege for access
- cc: cache control

  0: cached, 1: coherent, 2: noallocate 3:physical
- cs: coherence state

  4: read, 2: write, 1: replace
- p: priority, d: detail, s: sequential

Exceptions and Events

- Exceptions post events
- Events handled via minimal context switch
  - program counter and general register saved in D memory
  - Multiple events remain queued in event register
  - program counter & general register loaded from D memory
- Memory-mapped resources
  - Event register
  - Suspended thread's program counter & general register
- Precise exceptions, never masked

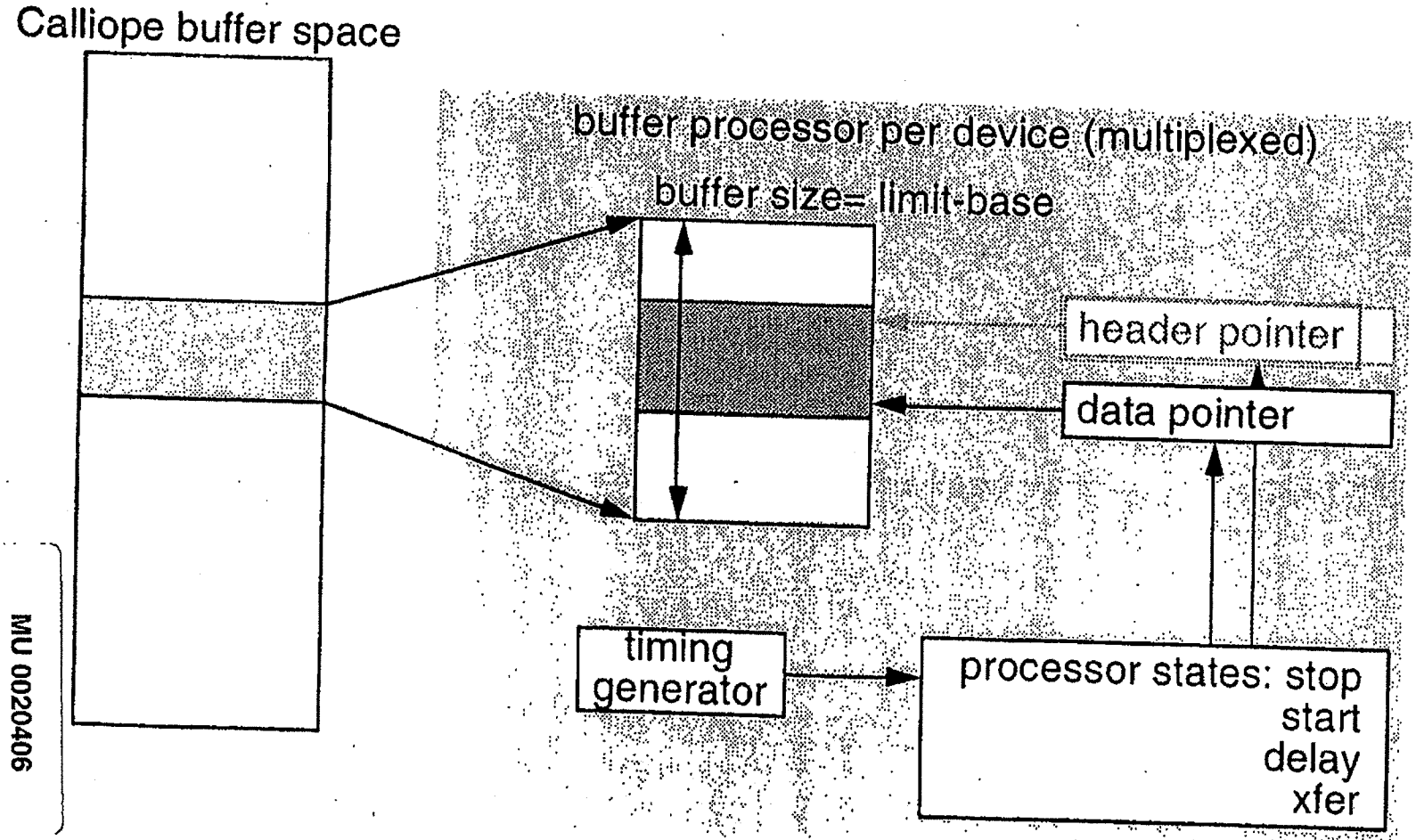
microunity

I/O structure

- Data moved by loads & stores (no DMA)
- Movement via event thread
- External interface chips - "Calliope"
  - buffer memory
  - buffer processor
  - timing generator
  - device formatters
  - device-specific interfaces

microunity

Calliope buffer memory and processor



Summary

- Full 64-bit general-purpose architecture
- Gigaflop supercomputer performance
- DSP capable of video and audio
- Powerful and flexible Gigabit I/O system

Mux operations viewed as functions on bit indices

- An arbitrary mux operation may be viewed as a function on the bit index:

$$\text{dest}[i] \leftarrow \text{src}[f(i)]$$

- The number of high index bits preserved by the function determines the "outer" group size.
- The number of low index bits preserved by the function determines the "inner" group size.

- For a 128-bit datapath, a bit index is 7 bits wide. If we preserve 2 high-order index bits, then we are operating on 4 groups of 32 bits. If we further preserve 3 low-order index bits, then we are operating on 8-bit groups within each 32-bit group.

This corresponds to an "outer" group size of 32 and an "inner" group size of 8.

- A "copy" operation, on bits, pecks, nibbles, etc., corresponds to setting a consecutive sequence of index bits to constant values.
- A reversal, or "swap", operation on bits, pecks, nibbles, etc., corresponds to complementing a consecutive sequence of index bits.
- A rotate operation corresponds to performing modular addition on a consecutive sequence of index bits.
  - — Zero fill and sign extend can be achieved through minor modifications of this.
  - — Expand and compress operations can be achieved by additionally performing right or left shifts on the high-order index bits.

- A shuffle/deal operation corresponds to performing a rotation on a consecutive sequence of index bits.
  - Viewed as any power-of-two rectangular matrix, a transpose of that matrix corresponds to a perfect shuffle/deal of some order.
  - Viewed as any power-of-two n-dimensional rectangle, an arbitrary transposition of the dimensions corresponds to a permutation on a consecutive sequence of index bits. Although it is possible to implement this generality, the encoding is somewhat cumbersome and requires too many bits to fit in an immediate.
- All of these functions on bit indices seem fairly easy to compute. However, a full crossbar for performing the data muxing is expensive to build. Is there a cheaper way?

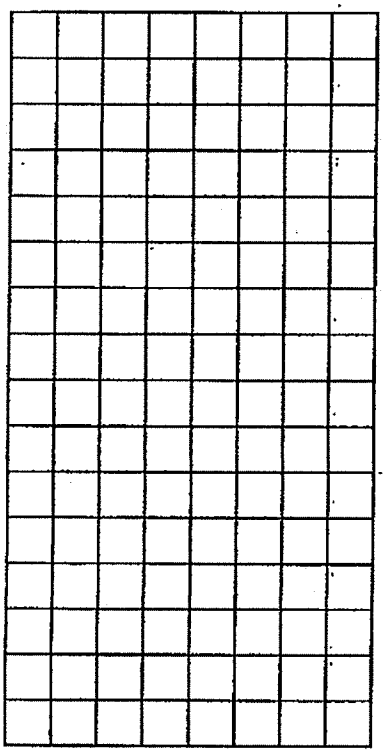
General permutation algorithms

- It can be shown that an arbitrary permutation of W bits can be performed by first arranging the data in an n-dimensional rectangle whose sides correspond to the factors of W. The permutation can then be achieved by performing a sequence of independent permutations along each dimension, followed by a second sequence of independent permutations which follows the dimensions in the opposite order, i.e., d1, d2, ..., dn, dn-1, ... d1. This is a sequence of 2*n-1 permutations.

- The case we're interested in is the 2-dimensional case. When arranged as a rectangle, an arbitrary permutation can be achieved by performing the following sequence of operations:
  a. Perform a set of independent row permutations on the data.
  b. Perform a set of independent column permutations on the data.
  c. Perform a set of independent row permutations on the data.
- If the row and column permutation operations are replaced with mux operations, some copying may also be achieved (although not all cases can be handled).

The XLU datapath

- Since our machine datapath is 128 bits wide, we are building a permutation network based on a 16 x 8 rectangle:



16 rows

8 columns

- Data enters along the rows. Each row has 8 data buses.
- Stage 1 consists of performing an 8:1 mux operation on each bit from the 8 data buses in its row. The results are placed on a set of data buses which run along the columns, with 16 buses per column.
- Stage 2 consists of performing a 16:1 mux operation on each bit from the 16 data buses in its column. The results are placed on a set of data buses which run along the the rows, with 8 buses per row,
- Stage 3 consists of performing an 8:1 mux operation on each bit from the 8 data buses in its row. Data leaves along the rows.

XLU datapath control

- Since each bit has two 8:1 and one 16:1 mux operations performed on it, we would need 128 * (2*3 + 4) encoded mux selects to perform all of these operations in the obvious way. This is 1280 independent mux controls. This seems like too much control logic and wiring.
- We can improve on this by generating multiple sets of control signals which are shared along columns (stages 1 & 3) or rows (stage 2), and a set of control selects which is shared along rows (stages 1 & 3) or columns (stage 2).

- The control for each bit is generated locally by performing an independent mux operation on each of the control bits being shared by that row or column.

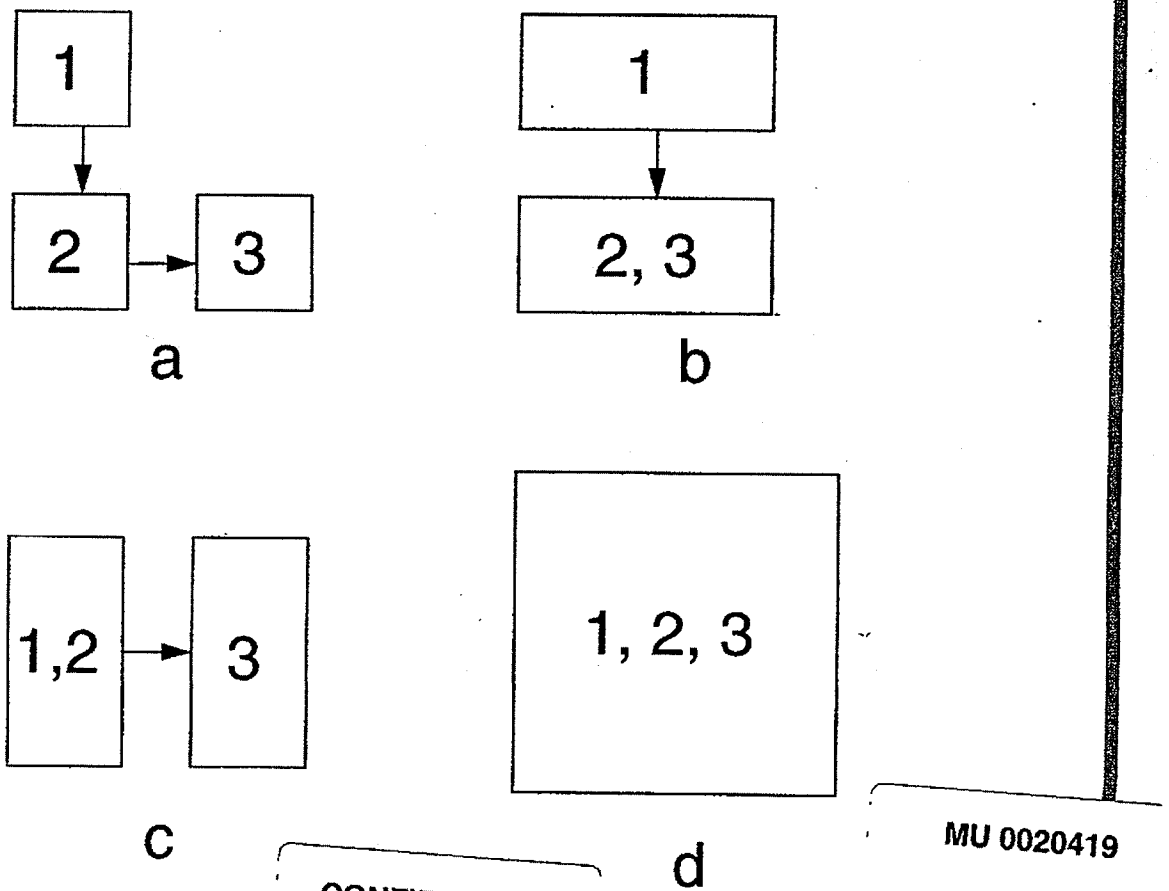
  For example, for a given bit in stage 1, there are two 3-bit shared control buses in its column, and a 3-bit shared control select bus in its row. Each of the 3 control select bits selects one of the two corresponding control bits. The resulting 3-bit value is then decoded and used to control the 8:1 mux for that bit.

- This breakdown of the XLU control into shared row and column signals significantly reduces the amount of control logic and wiring for the XLU.

XLU placement and routing

- It is initially intuitive to think of the stage 1, 2, and 3 muxes for a given bit being in close physical proximity to one another. However, this is not necessary. The data flow from stage 1 to stage 2 is along columns, so the stage 1 and stage 2 muxes for a given bit must be in the same column. The data flow from stage 2 to stage 3 is along rows, so the stage 2 and stage 3 muxes for a given bit must be in the same row. This still leaves room for four basic placement strategies.

- Both a and b have the undesirable property that the data coming in and out isn't aligned with the rest of the datapath.
- Placement c has the advantage that some of the row wires don't need to coexist as they would in d. This is the placement we are using.



XLU functional control

- The shared row and column control signals for the XLU are generated by several indepenent control modules, each of which is specific to a particular class of operations. For example, shuffle control, shift/rotate control, copy/swap control, etc.
- These control signals are then selected by a mux operation. The outputs of these mux operations are the shared row and column control signals used for the XLU datapath control.

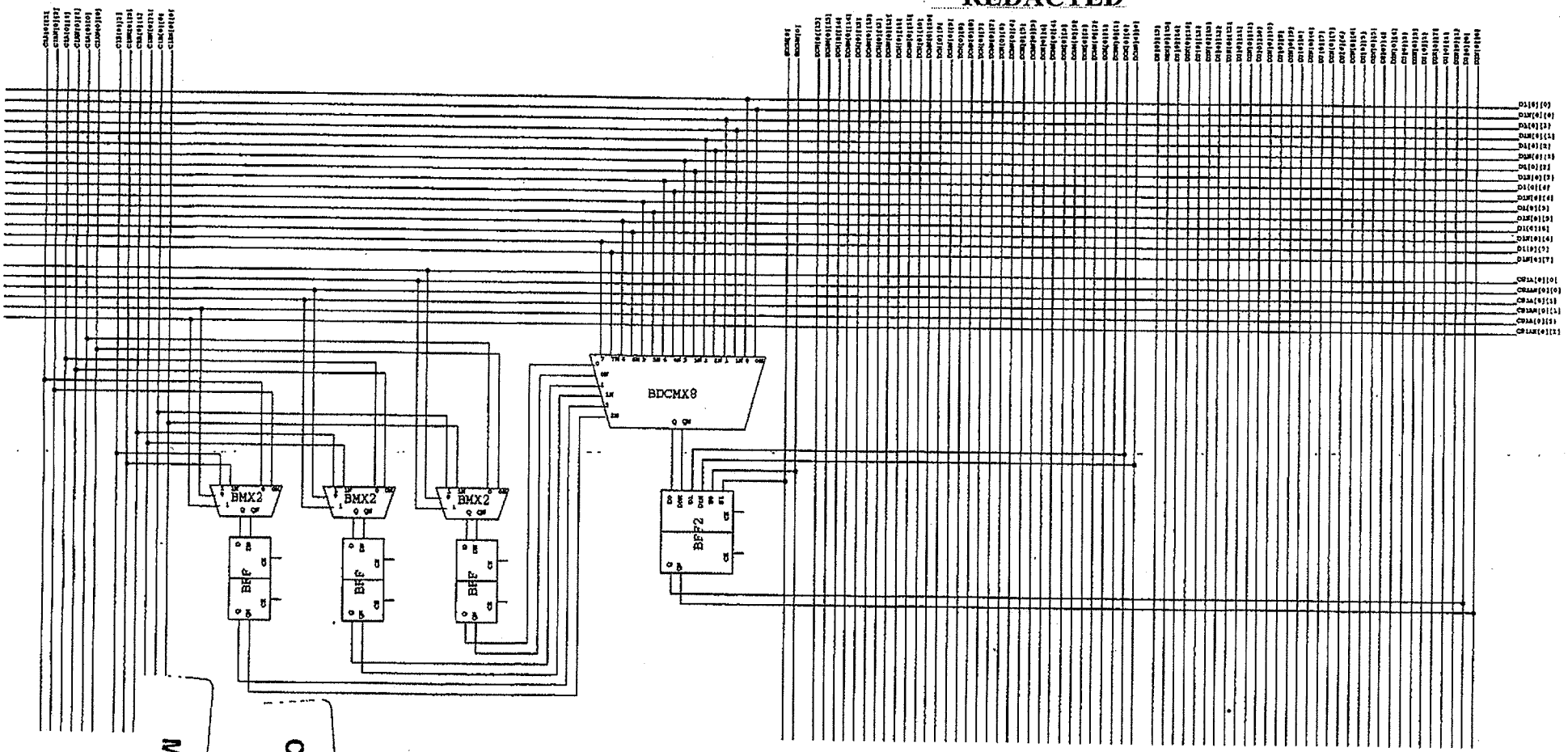
Random little details

- The XLU also performs a load alignment function. This function bypasses the normal stage 1 mux operation, and is instead muxed into the datapath at the end of stage 1.
- While stages 1 and 2 use two sets of control signals, stage 3 uses three in order to handle sign extension. In addition, an addional pair of control signals is used to implement the shufflemux family of intructions. These additional control buses are shared between the high and low 64 bits of the datapath.

- Stage 3 also performs zero/merge fill for some of the shift-related instructions. This is achieved by selecting shared column control signals with shared row selects, and using the result to determine whether the result should be taken from the main datapath or the fill bus.

SIds
Confidential information of MicroUnity. Provided pursuant to REDACTED agreement

TITLE: BS1 (BIT 0)
ENGINEER: TJK
DATE: LAST_MODIFIED= REDACTED
REV# NOABBREV

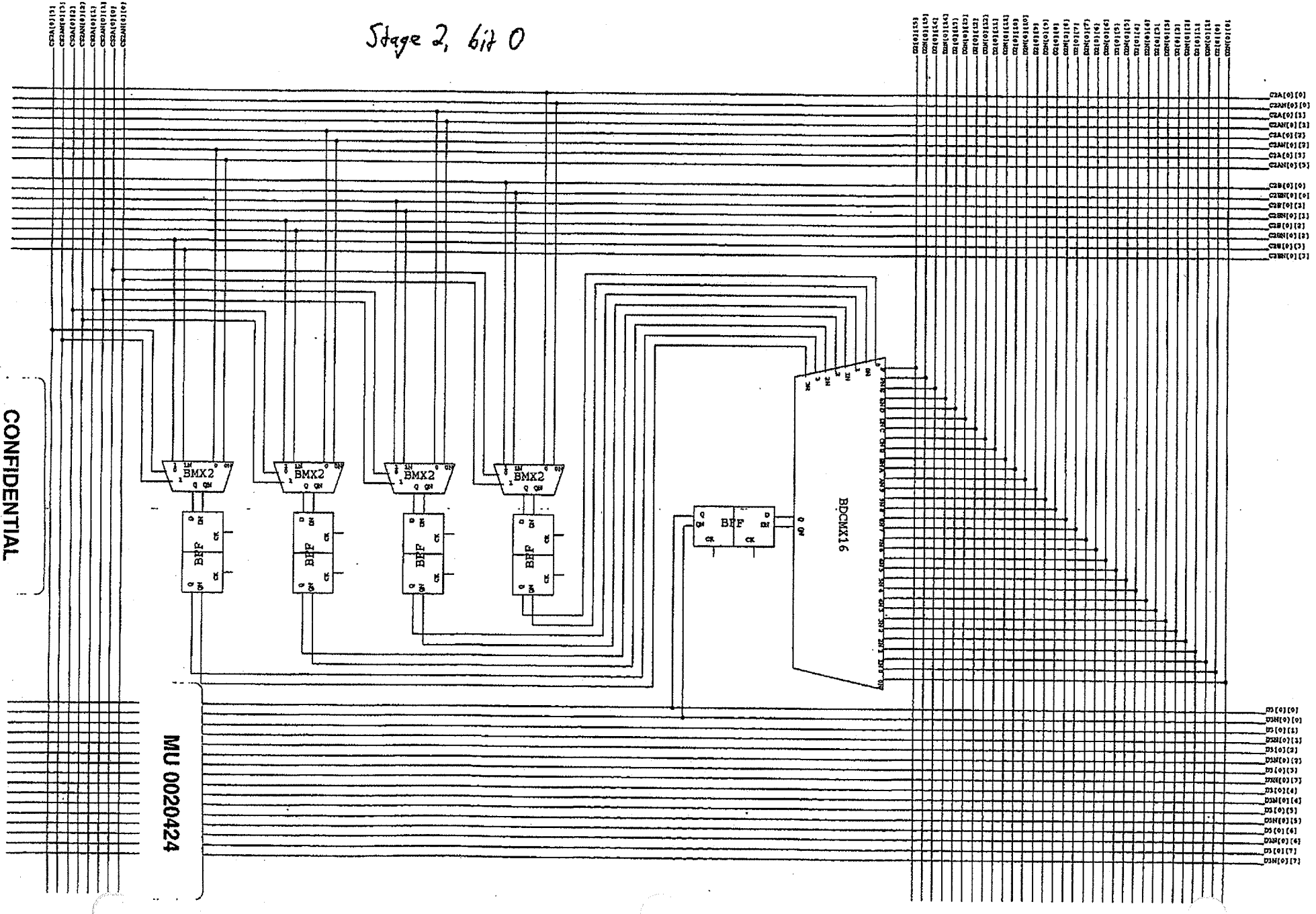


Stage 1, bit 0

Id
Confidential information of MicroUnity. Provided pursuant to REDACTED agreement

TITLE: BS2 (BIT 0)
ENGINEER: TJK
DATE: LAST_MODIFIED= REDACTED
REV# NOABBREV

Stage 2, bit 0



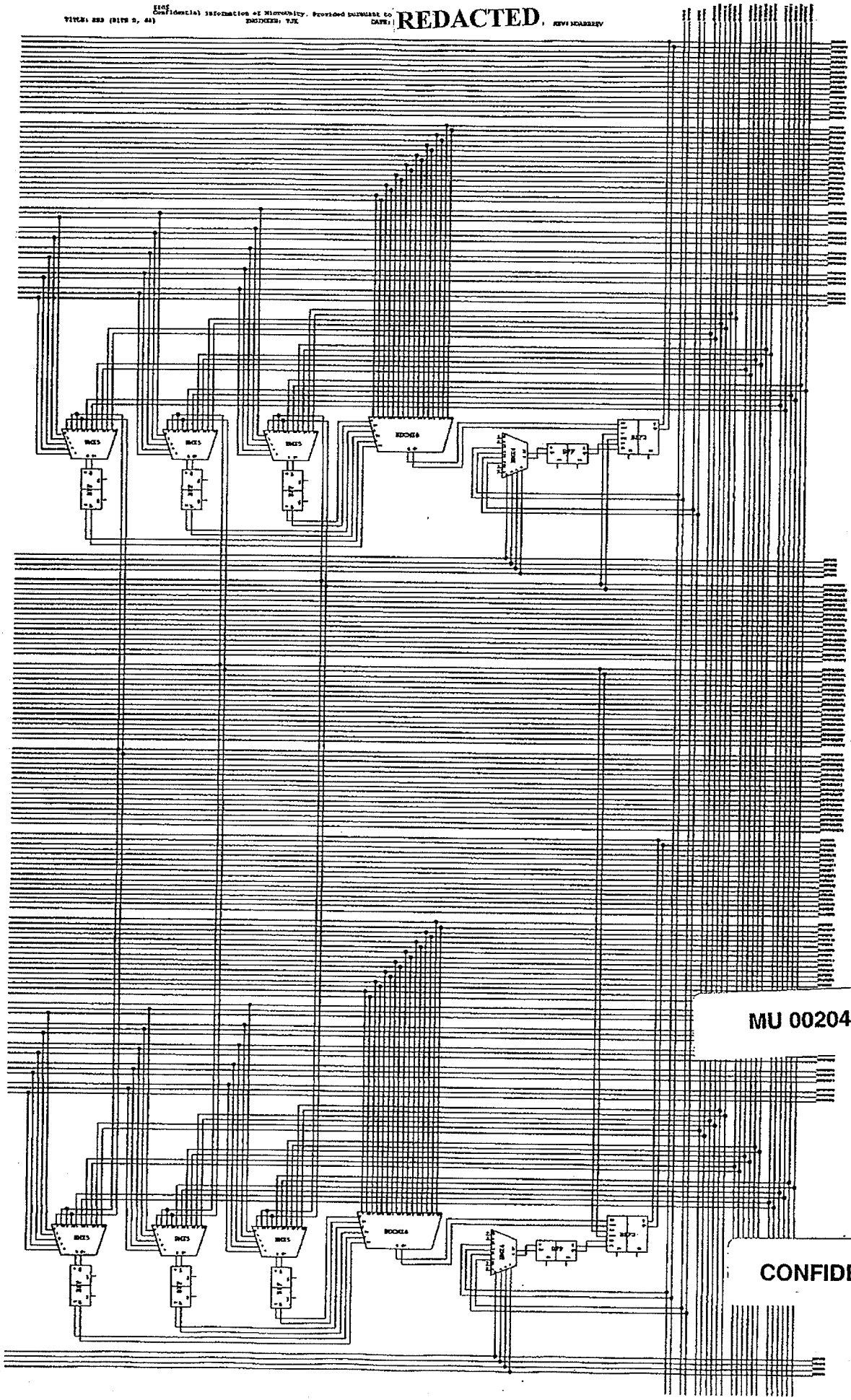
CONFIDENTIAL

MU 0020424

Stage 3

bit 0

bit 64

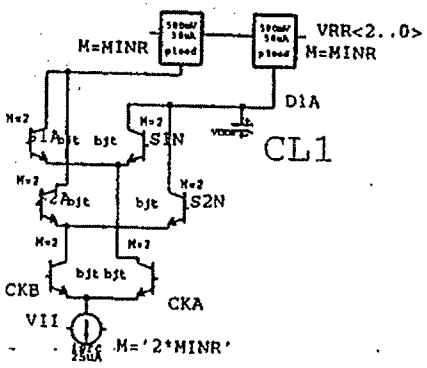


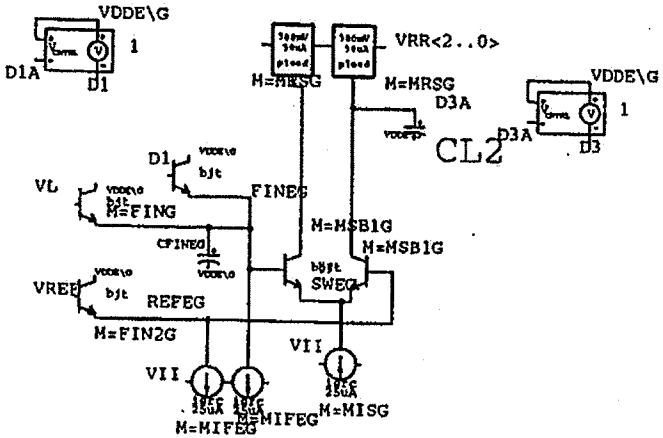
TITLE: XPATH

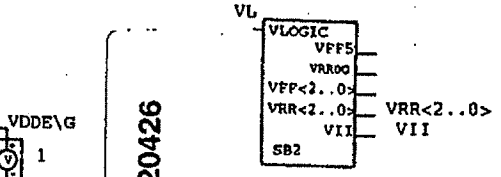
ENGINEER: WHH REDACTED DATE:

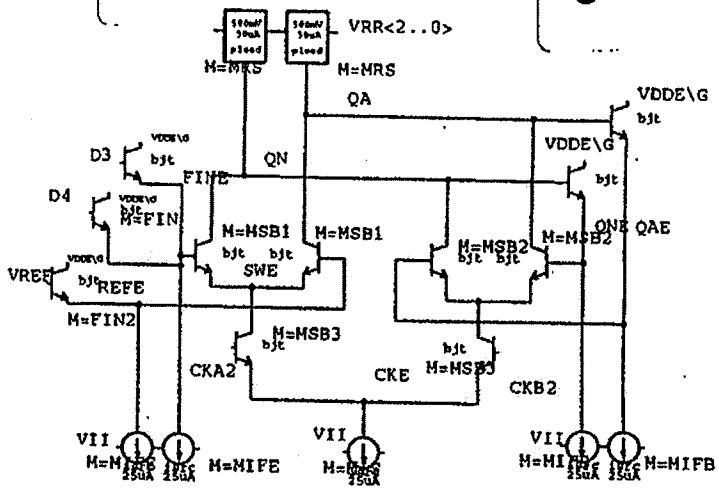
REVI NOABBREV

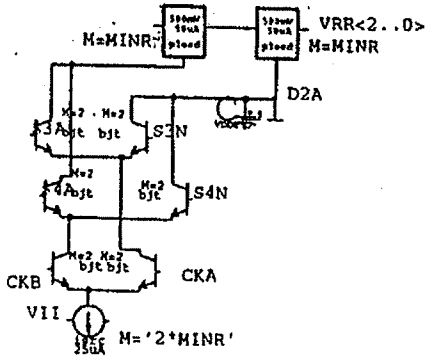
LAST_MODIFIED: REDACTED

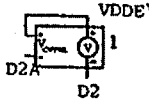


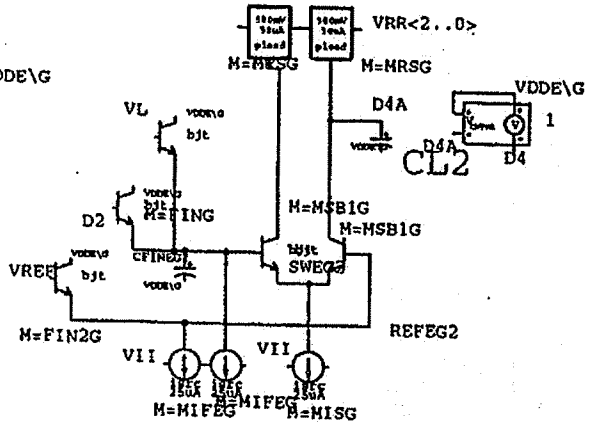


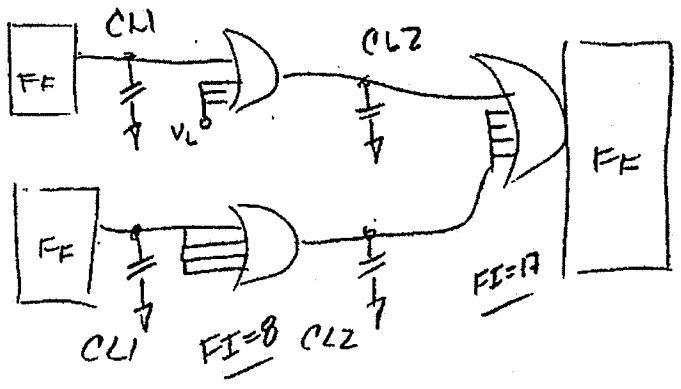


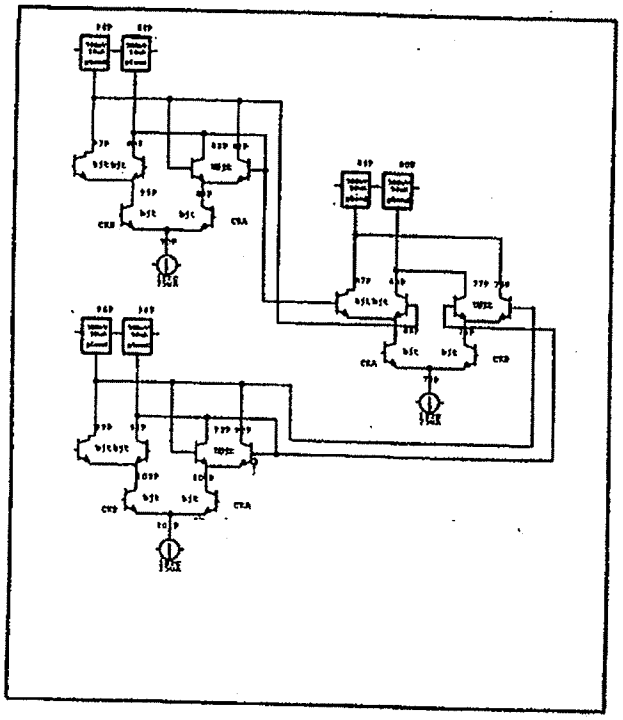




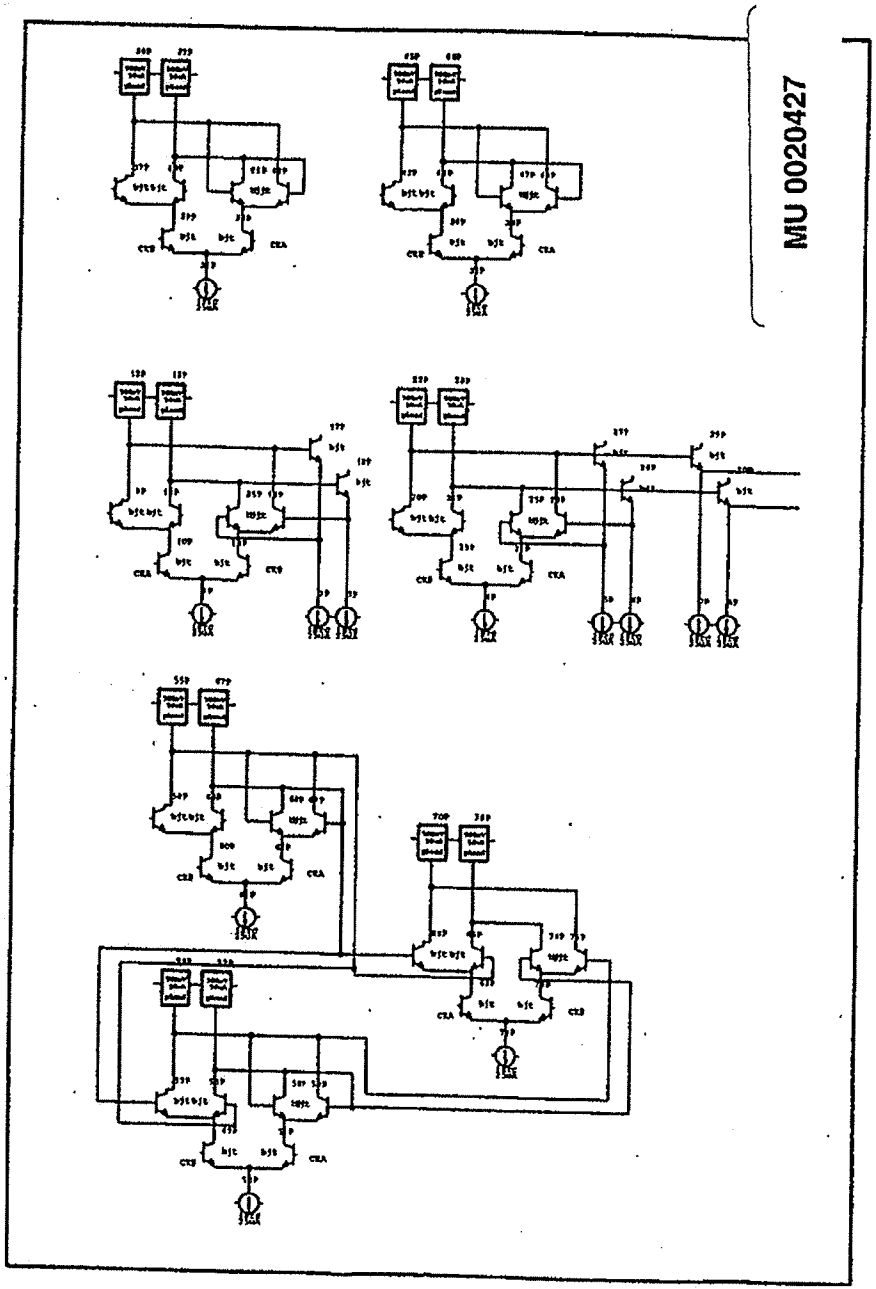








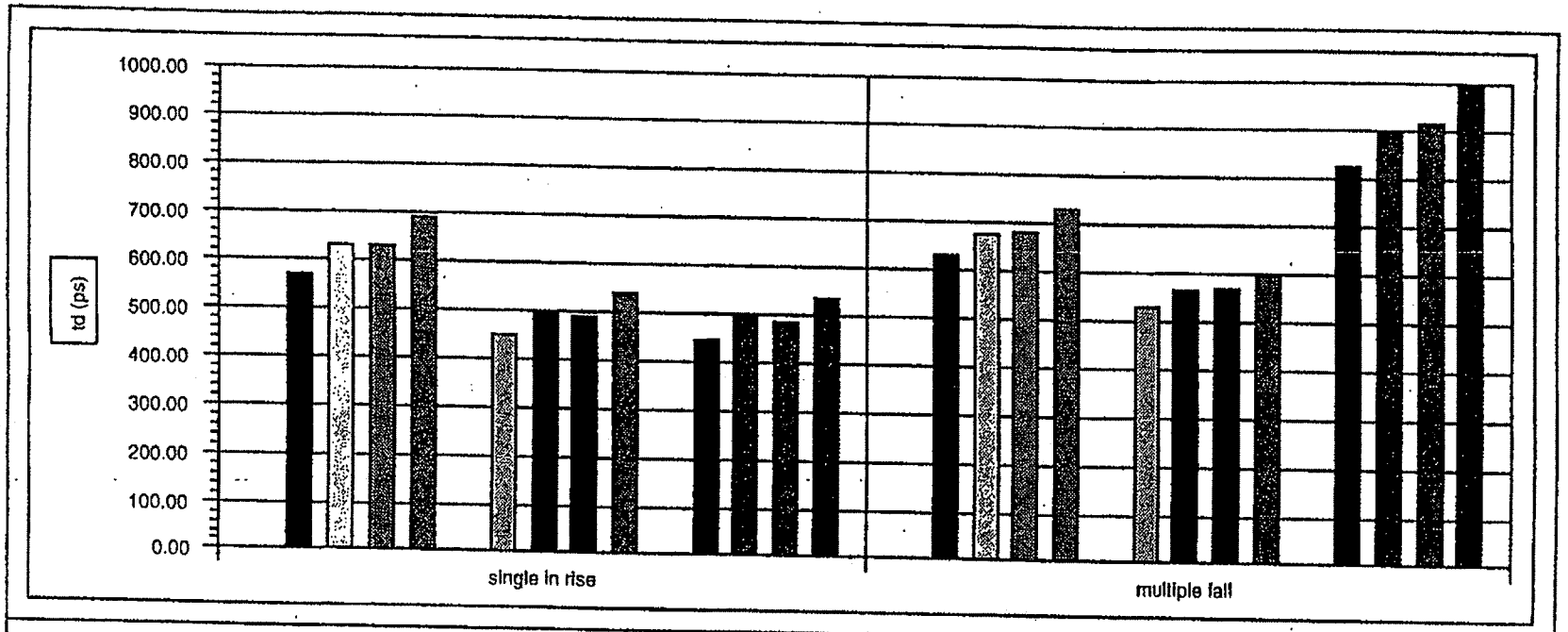
FF CHANGES EVERY 800PS
1600PS PERIOD CLOCK



FF CHANGES EVERY 800PS
~~1600PS~~ 800 PERIOD CLOCK

MU 0020427

CONFIDENTIAL



		td (ps)		rf=400mv/400ua=1k									
	cond	single in rise	multiple fall	dt1/dc1 (ps/ff)	dt2/dc1	dt1/dc2	dt2/dc2	c1	c2	fin	fin2	fing	fin2g
comp fi=8/16	1	569	628	1.23	0.96	1.19	1.04	49ff	49ff	16	4	8	2
	2	630	676	1.23	0.98	1.19	1.06	99ff	49ff	16	4	8	2
	3	628	680					49ff	99ff	16	4	8	2
	4	690	729					99ff	99ff	16	4	8	2
no comp fi=1/1	5	447	530	0.94	0.63	0.85	0.76	49ff	49ff	1	1	1	1
	6	494	562	0.95	0.64	0.86	0.78	99ff	49ff	1	1	1	1
	7	489	569					49ff	99ff	1	1	1	1
	8	537	601					99ff	99ff	1	1	1	1
o comp fi=8/1(	9	444	823	0.95	1.55	0.83	1.84	49ff	49ff	16	1	8	1
	10	491	900	0.96	1.57	0.84	1.87	99ff	49ff	16	1	8	1
	11	485	915					49ff	99ff	16	1	8	1
	12	533	994					99ff	99ff	16	1	8	1

Circuit Speed/Power Optimization

- Motivation
- Timing-Driven Power Optimizer
- Simplified Delay Modelling
- Problems with Simple Models
- Improved Delay Modelling

Motivation

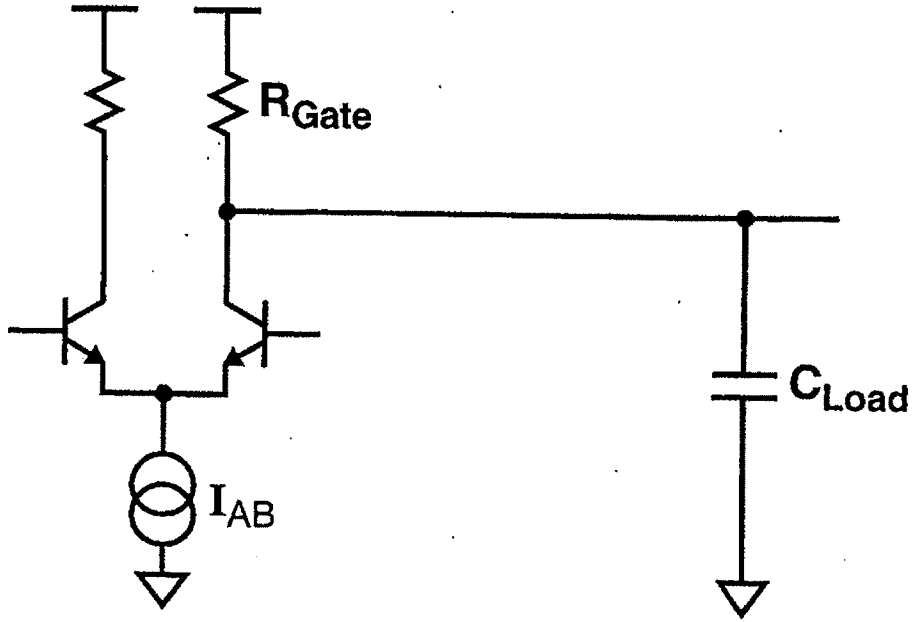
- Chip speed is limited by the slowest path
  - — Need tuned drive strengths to guarantee speed
- Power of constant-current circuits is proportional to drive strength
  - — Getting the most performance/Watt requires careful gate-level power tuning on a per-path basis
- Wire load dominates most nets and is indeterminate until after place & route
- Gate area is proportional to drive strength
  - — Need iterative speed power optimization

Timing-Driven Power Optimizer (topt)

Given a cycle time goal:

- Analyze the delay of every single-cycle path between flip-flops
- Determine the minimum-allowed signal level for unspecified paths
- Replace the gates in the path with ones which minimize area and power
- Try to make all paths critical!

Simplified Delay Modelling

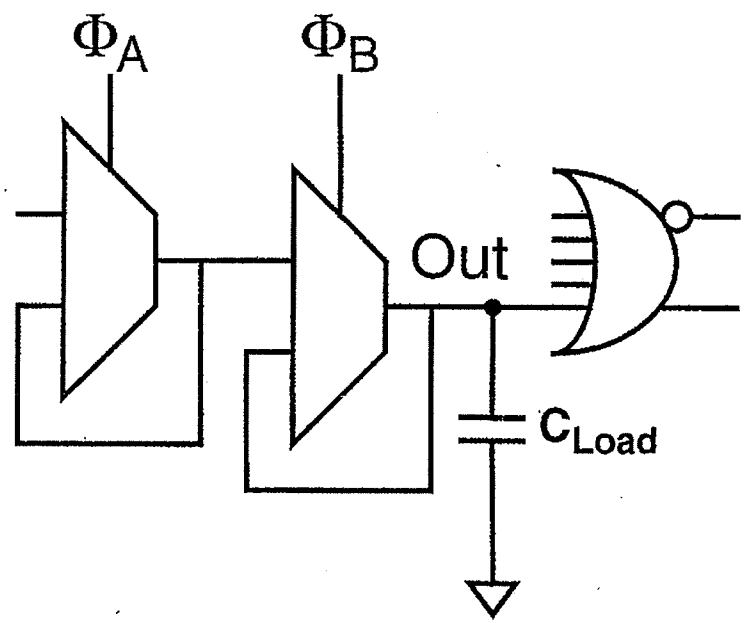


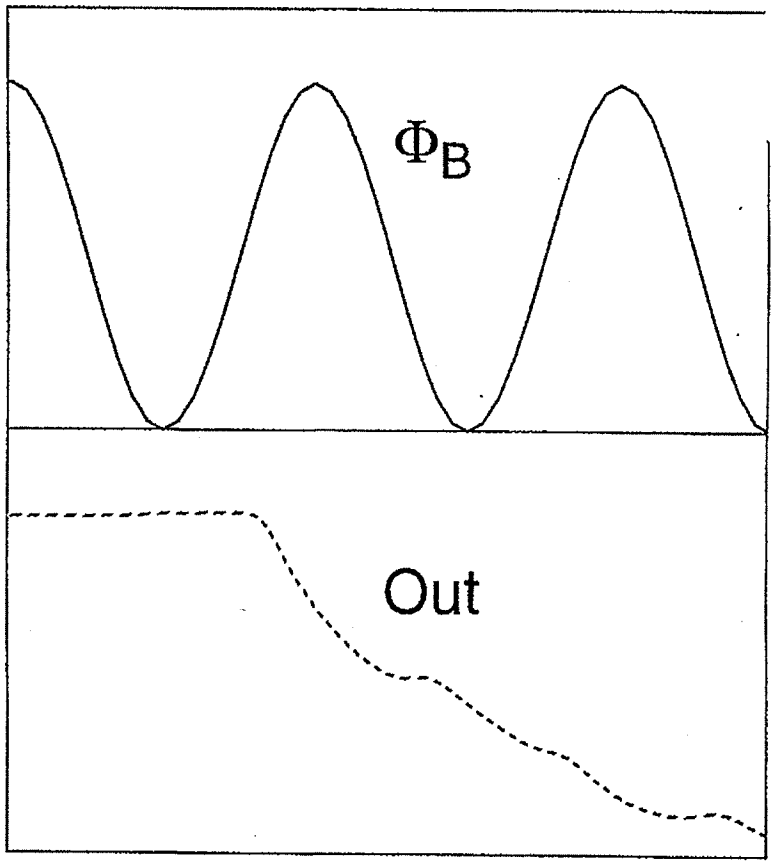
$$T_{Delay} = T_{Int} + \log(2)*R_{Gate}*C_{Load}$$

Problems with Simple Models

- Real gates are sensitive to input slope
  - — Many cell libraries forced to use worst-case (i.e. slowest) input slopes to guarantee performance by overpowering
- Poor input slope increases both T_{Int} and output slope
  - — 0.7*R*C doesn't tell the whole story of load dependence
- Slope-dependent effects much worse for some gates
  - — Wide *OR* gates with shifted references are especially bad
- Output waveforms do not always act like $e^{\frac{-t}{RC}}$
  - — Difficult to model slope with simple equations

Example - FF drives OR/NOR





What are the delay and slope of the flip-flop output?

topt Delay Model

Gate delay is table-derived function of

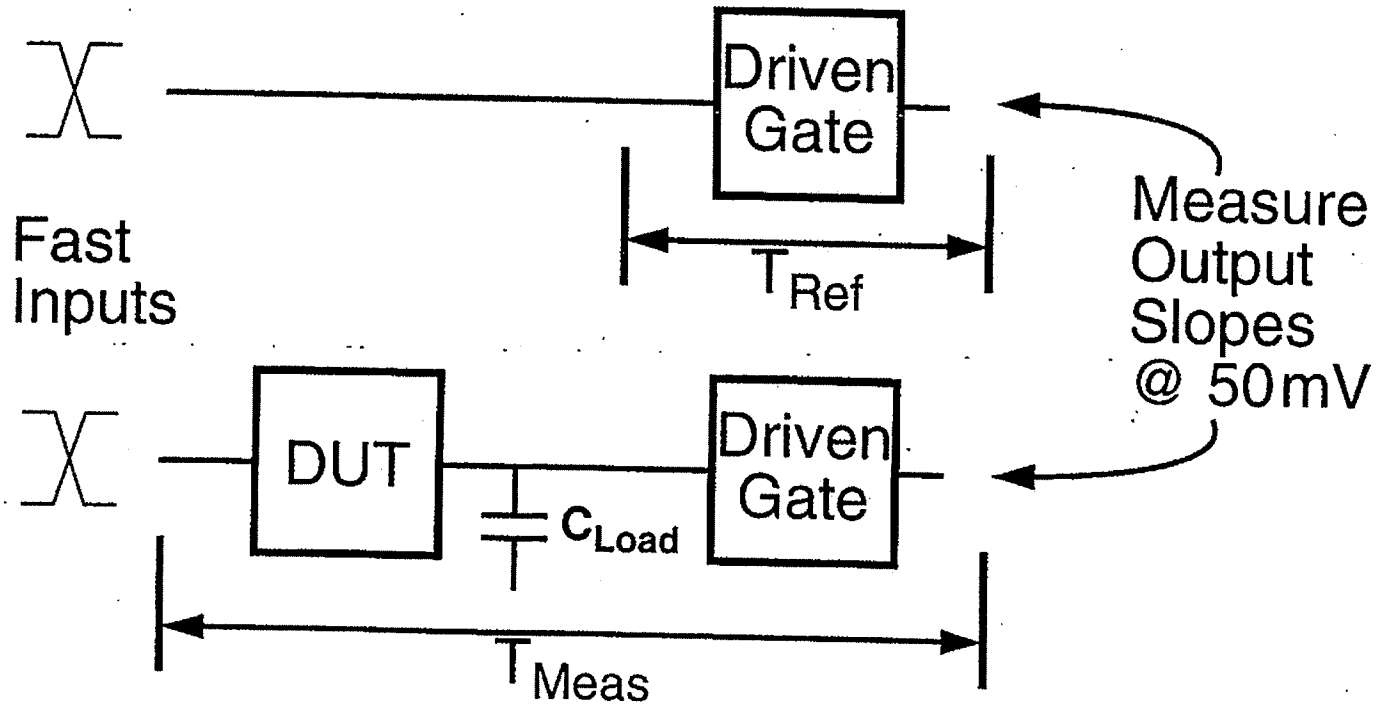
- Driving gate (i.e. which set of tables *topt* chooses)
- Output load capacitance
- Driven gate fanin (models impact of poor slope)
- Driven gate type (combinatorial or sequential)
  - — Flip-flops assumed not to pass bad slopes, but require larger input transitions to satisfy latching constraints

Lump all slope-dependent effects upon delay of driving gate, but attempt to model it in context as best as we can.

topt Delay Calculation

```
foreach net in path
   C_net = wire capacitance + Σ(gate input capacitance)
   Dly_tbl = f(driving_gate, driven_gate_type,
               driven_gate_fanin)
   Stage_dly = linear interpolation between delays of
               bracketing C_net entries in Dly_tbl
   Path_dly = Path_dly + Stage_dly
endfor
```

Delay Simulation



Insertion delay model:

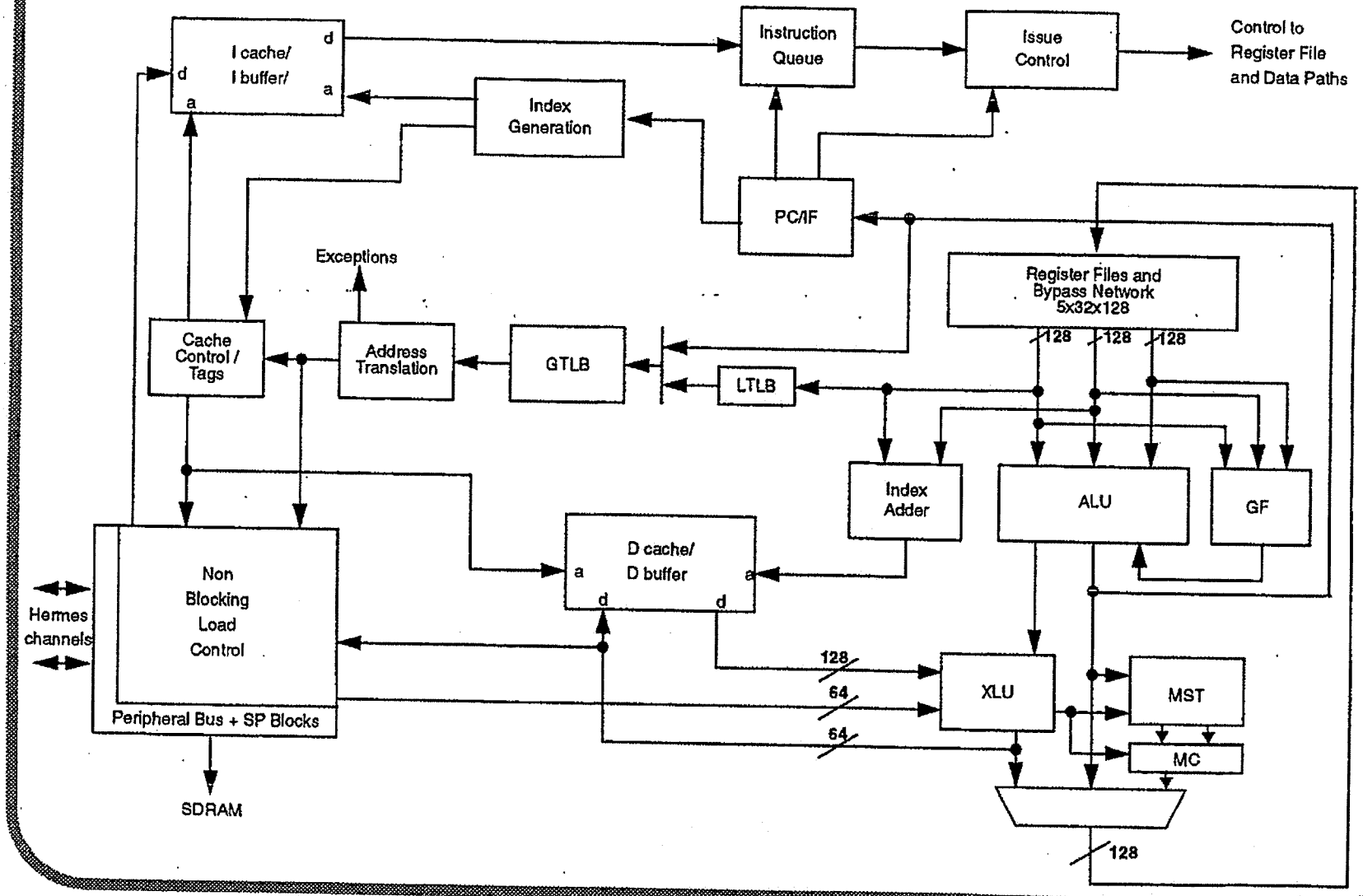
- $T_{Delay} = T_{Meas} - T_{Ref}$

Delay Model Complications

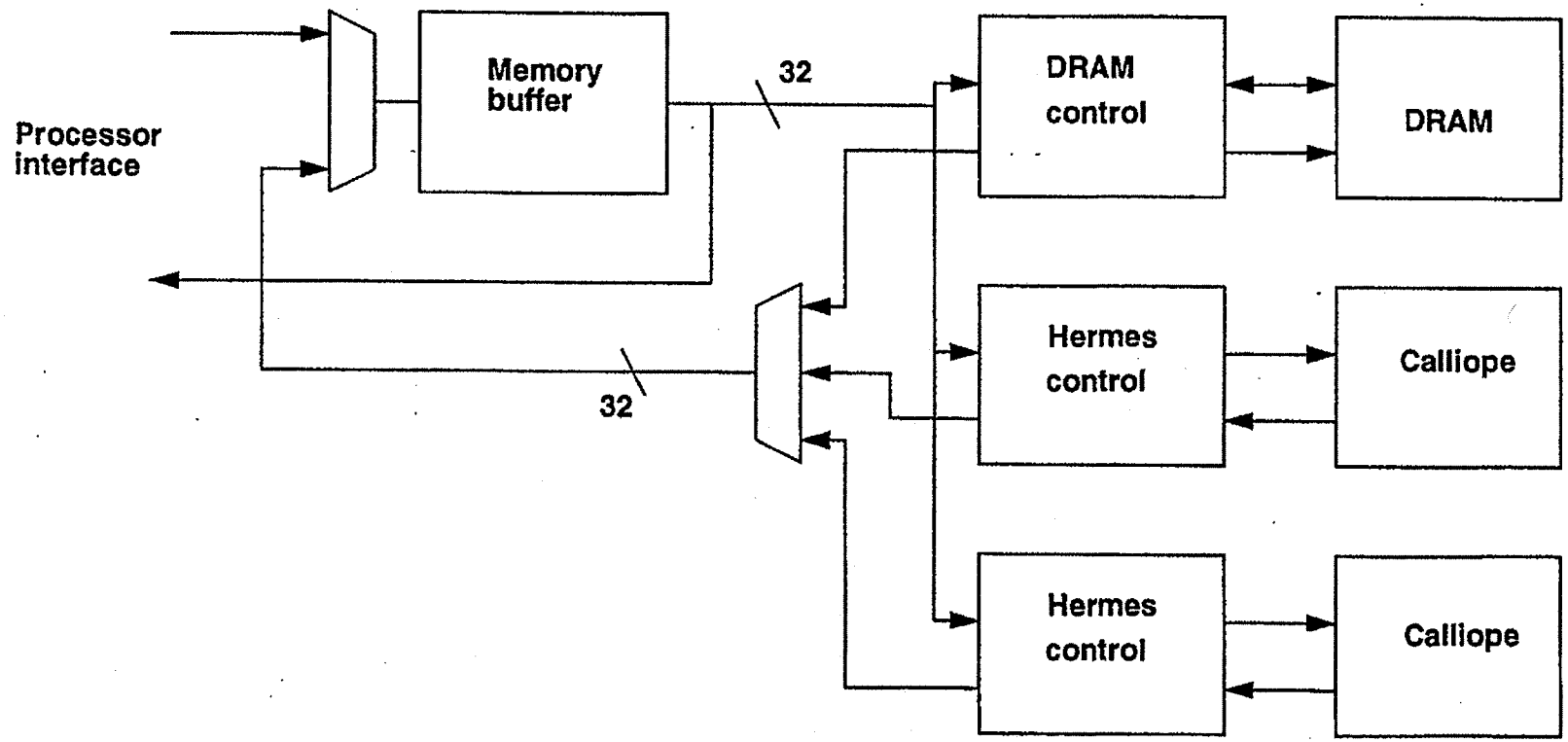
- If driven gate is combinatorial, add measured delay to compensate for slowing its output slope
- If driven gate is sequential, its (slave) output slope is assumed to be independent of input slope, but T_{Meas} and T_{Ref} measured to 50mV differential at the latch feedback nodes
- Due to similarity of many gates, there are relatively few driven-gate combinations to simulate

microunity

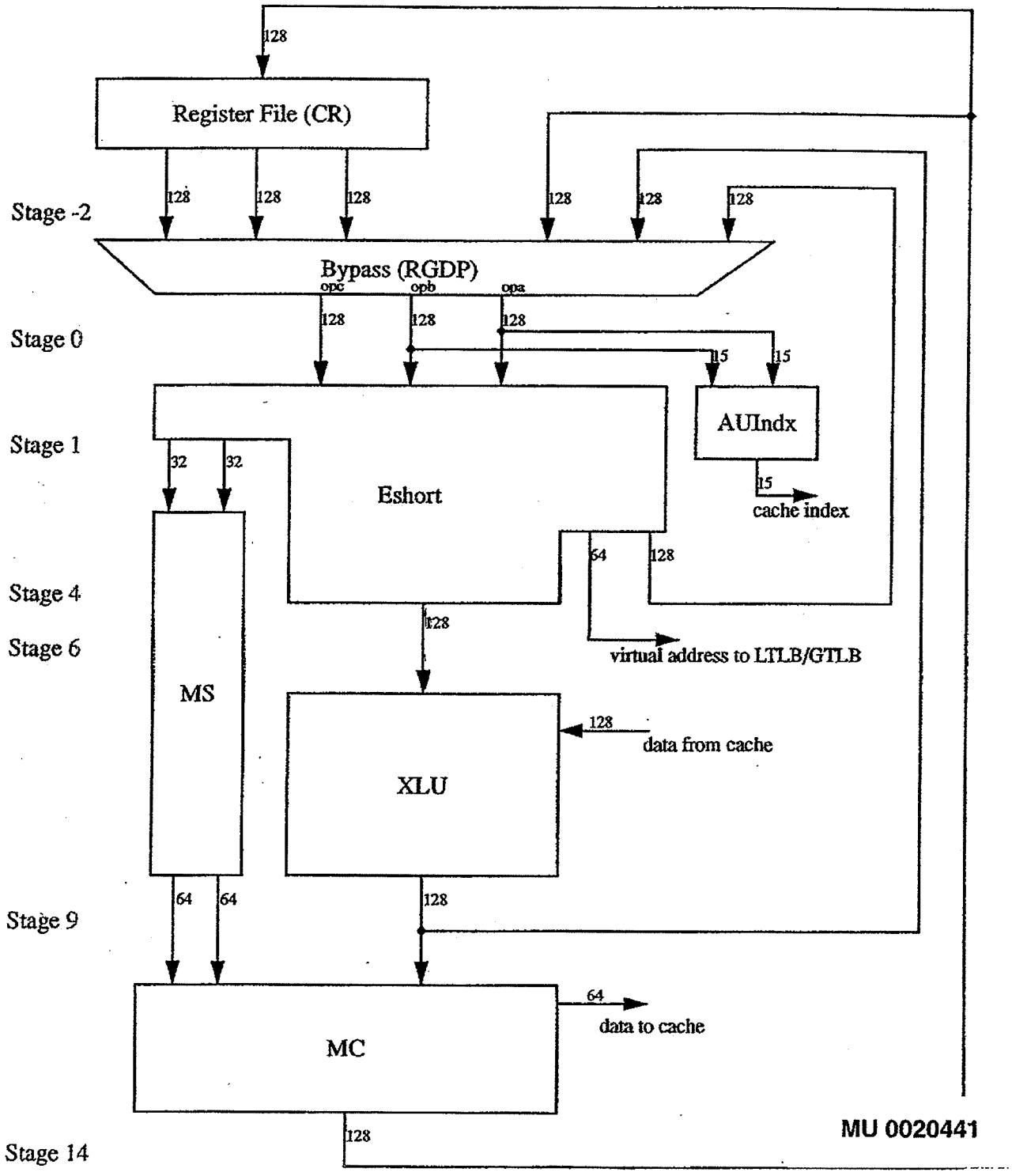
Euterpe Block Diagram



Non-blocking Load Buffer



Main Pipeline



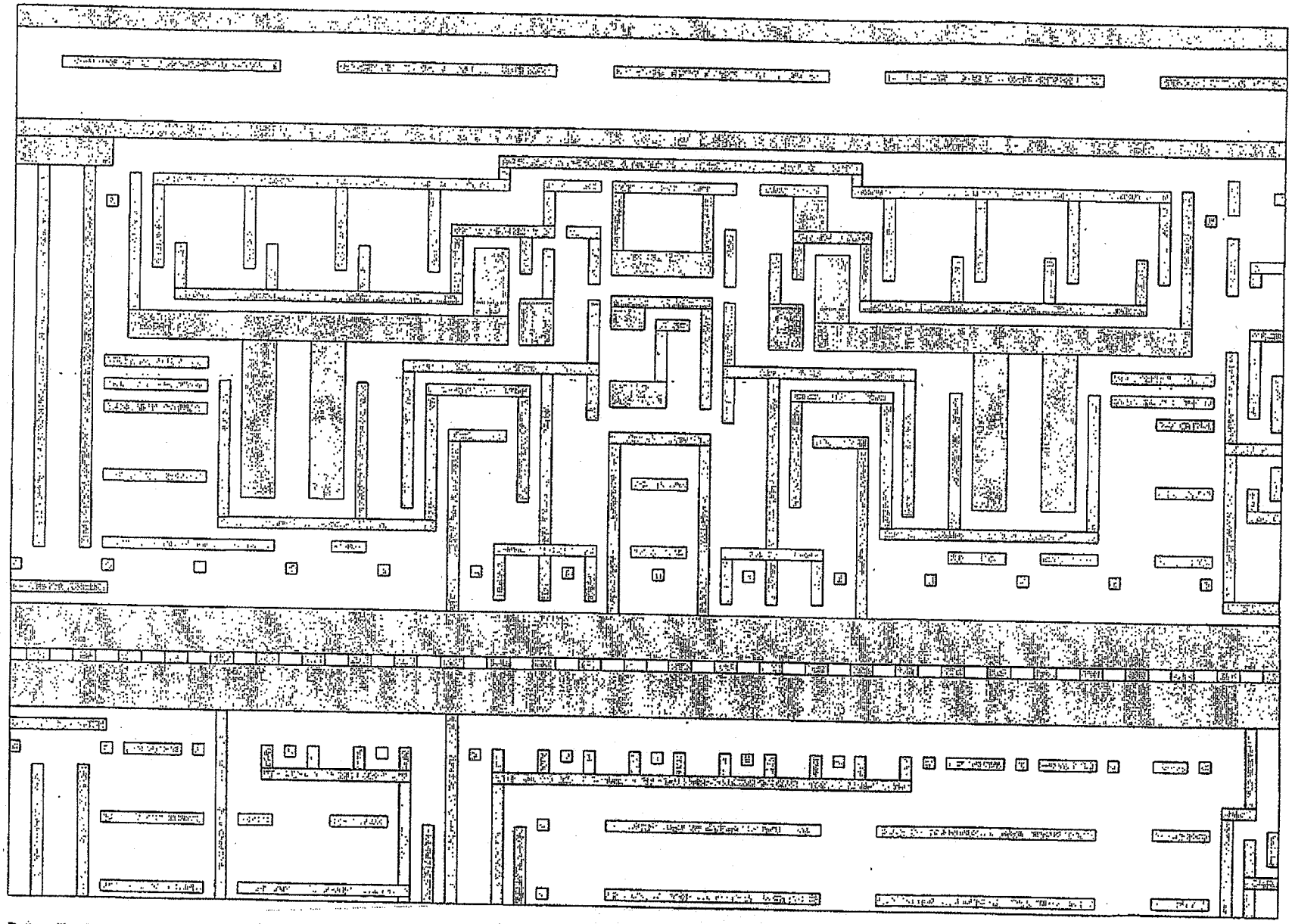
Instruction Pipeline

Mask Data Processing

- In-house tool, "vlsimm" used for all back-end mask data processing.
- Derived layer synthesis (uses geometric AND, OR, grow/shrink etc.)
- Wafflization & perforation of metal layers to regulate pattern density.
- Computation of airbridge support structures.
- DRC checking of all derived data.
- Computation of Optical Proximity Correction (OPC) features: serifs, scattering bars, anti-scattering bars.
- Application of mask-vendor-specific feature biases.
- Direct output of MEBES pattern format, with automatic arrayed figure compaction.
- Post-fracture readback XOR check of pattern data.
- Complete MEBES job deck synthesis: composite reticle contains scribe frame, die patterns, bar code, fiducial/alignment marks etc.

Mask Data Processing

- Typical 28-layer reticle set contains around 6 billion rectangles.
- Figure compaction often achieves < 2 bytes per rectangle (uncompacted MEBES is 8 bytes per rectangle minimum).
- Fracturing is run on a 4-CPU SGI Challenge machine with 2GB of physical memory. An entire mask set can be fractured (including post-fracture DRC & XOR checks) in 2-3 weeks.
- 68 production reticle tapes issued to date.
- In-house pattern file viewer, "mebesview" supports instant examination of fracture results, automatic overlay of DRC/XOR flags, "pushbutton" hardcopy on PostScript laser printer or Versatec plotter.
- Key constraint: "vlsimm" processes Manhattan rectangles only - internal algorithms are all vertex-based for maximum speed. Process design rules disallow non-Manhattan geometry on all layers.



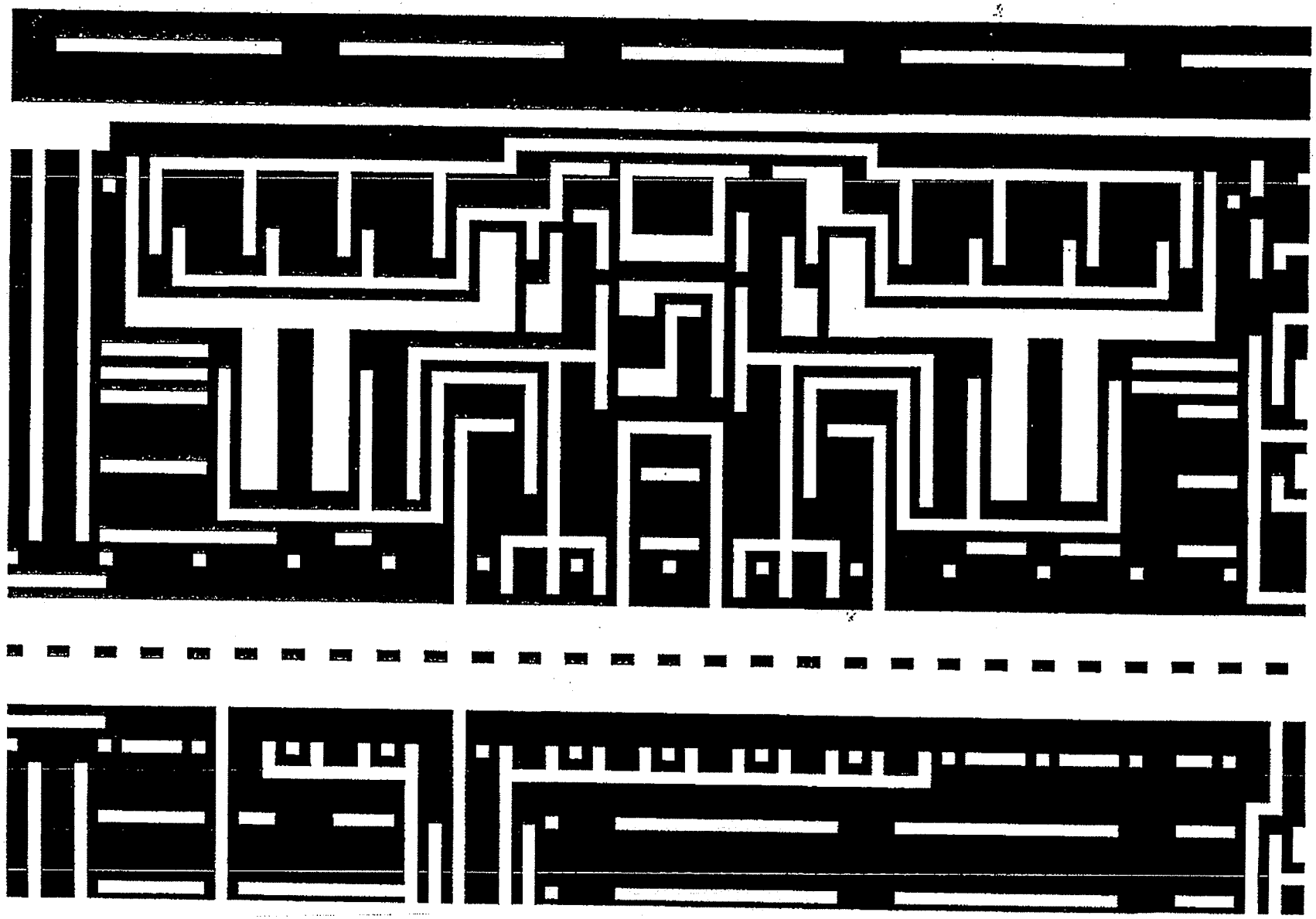
Pattern file: "199991800.03" Created:

REDACTED

MICROUNITY CONFIDENTIAL

MU 0020445

CONFIDENTIAL



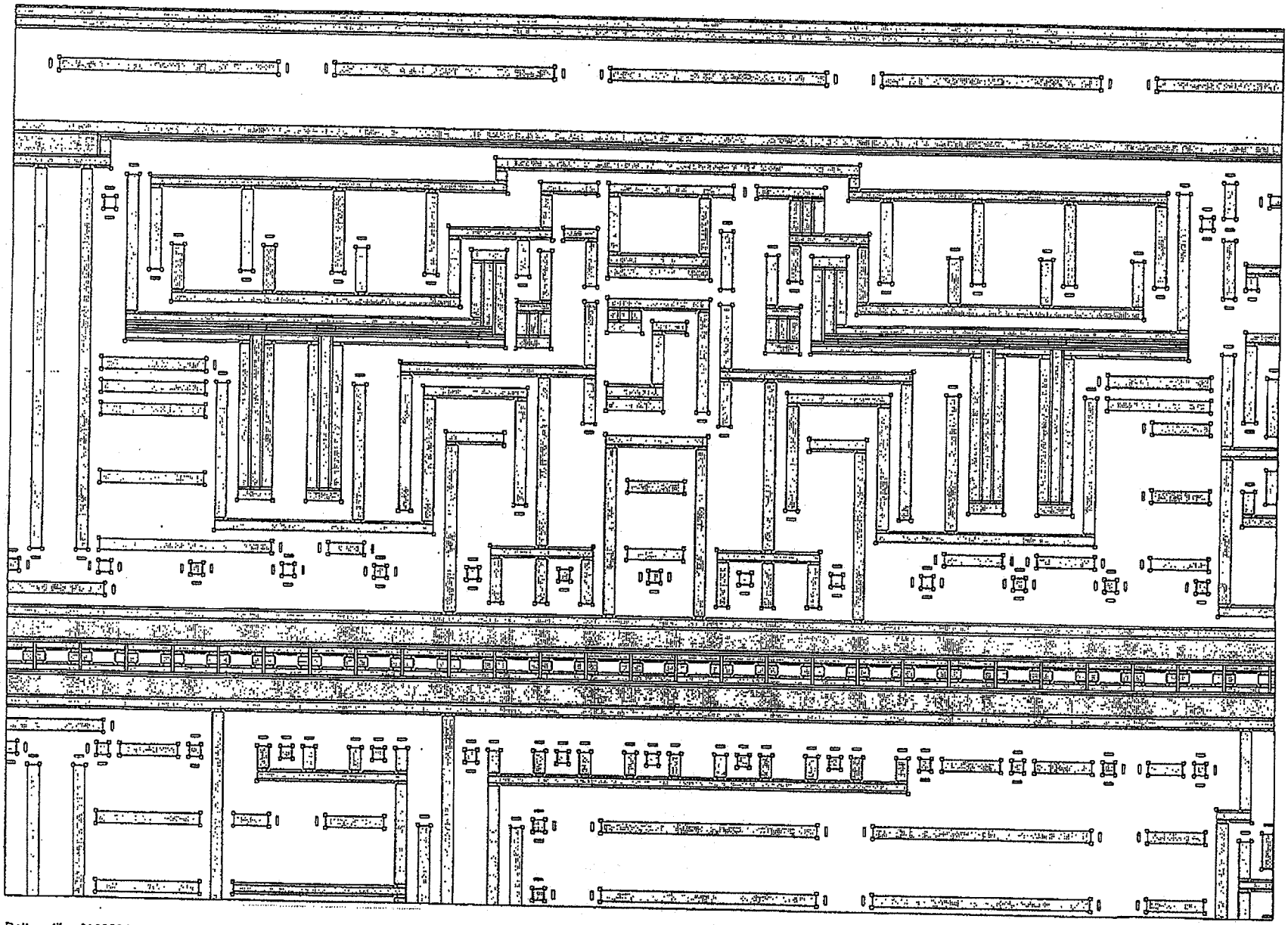
Pattern file: "199991800.03" Created:

REDACTED

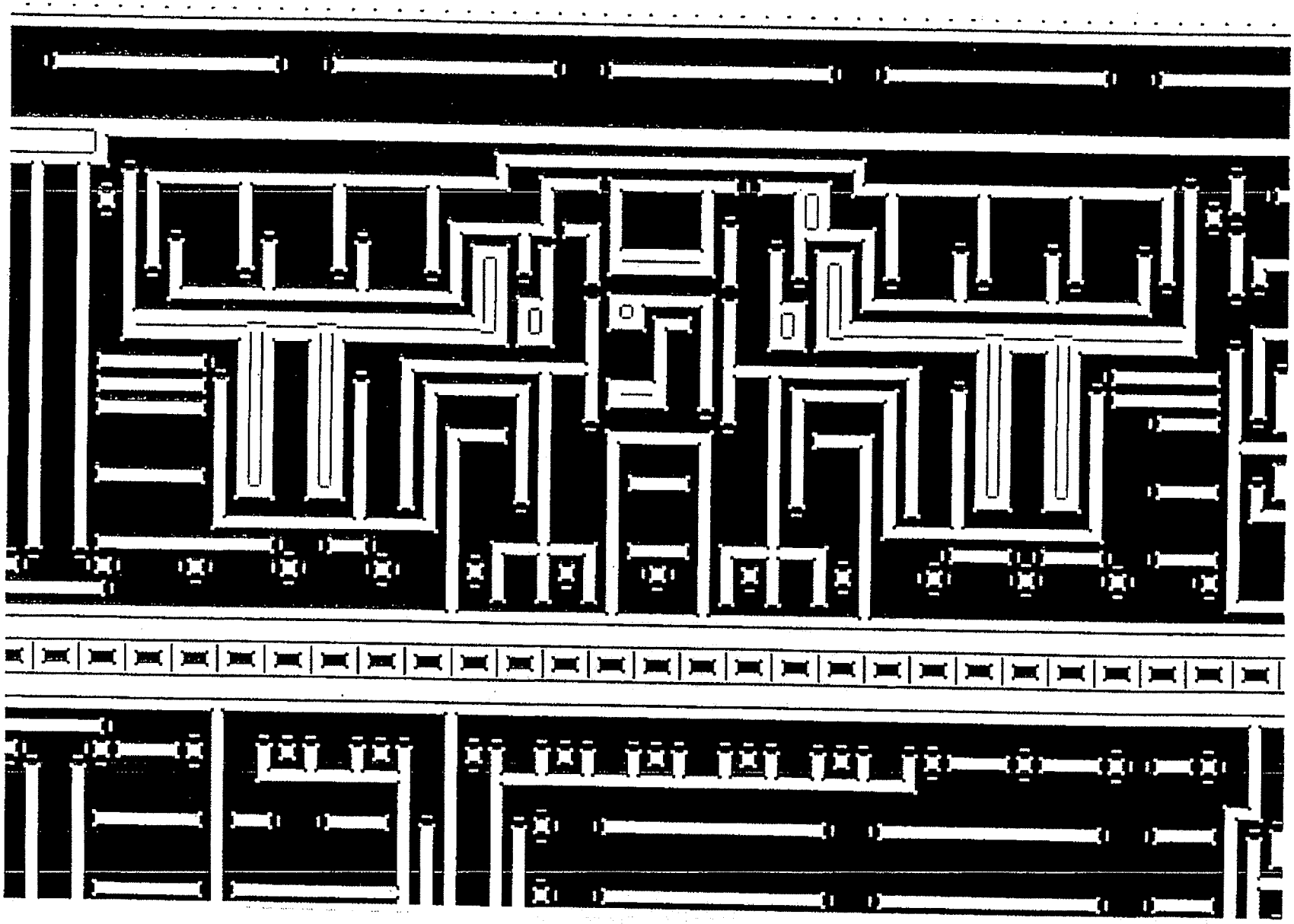
MICROUNITY CONFIDENTIAL

MU 0020446

CONFIDENTIAL



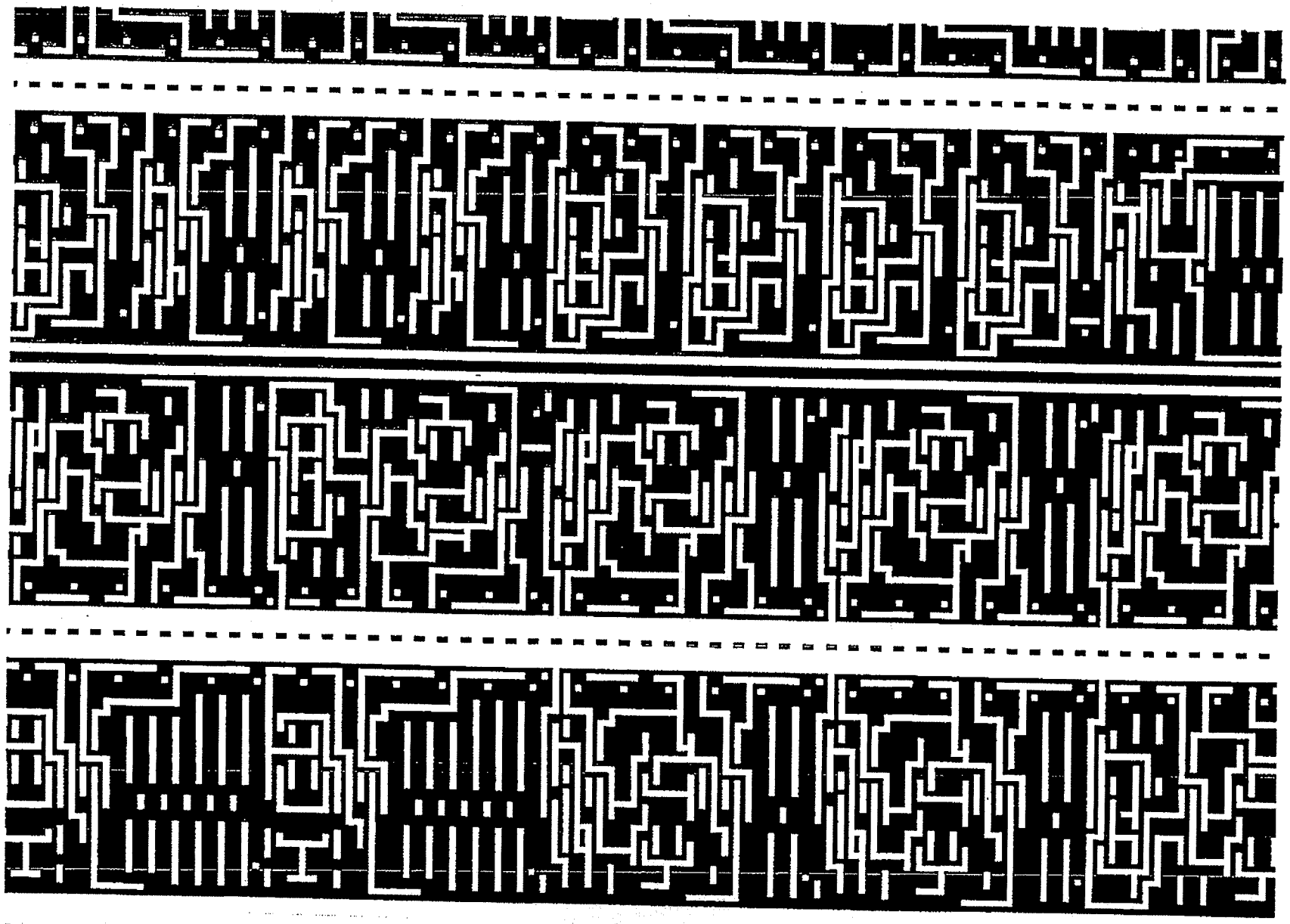
Pattern file: "199991800.03" Created: REDACTED



Pattern file: "199991800.03" Created:

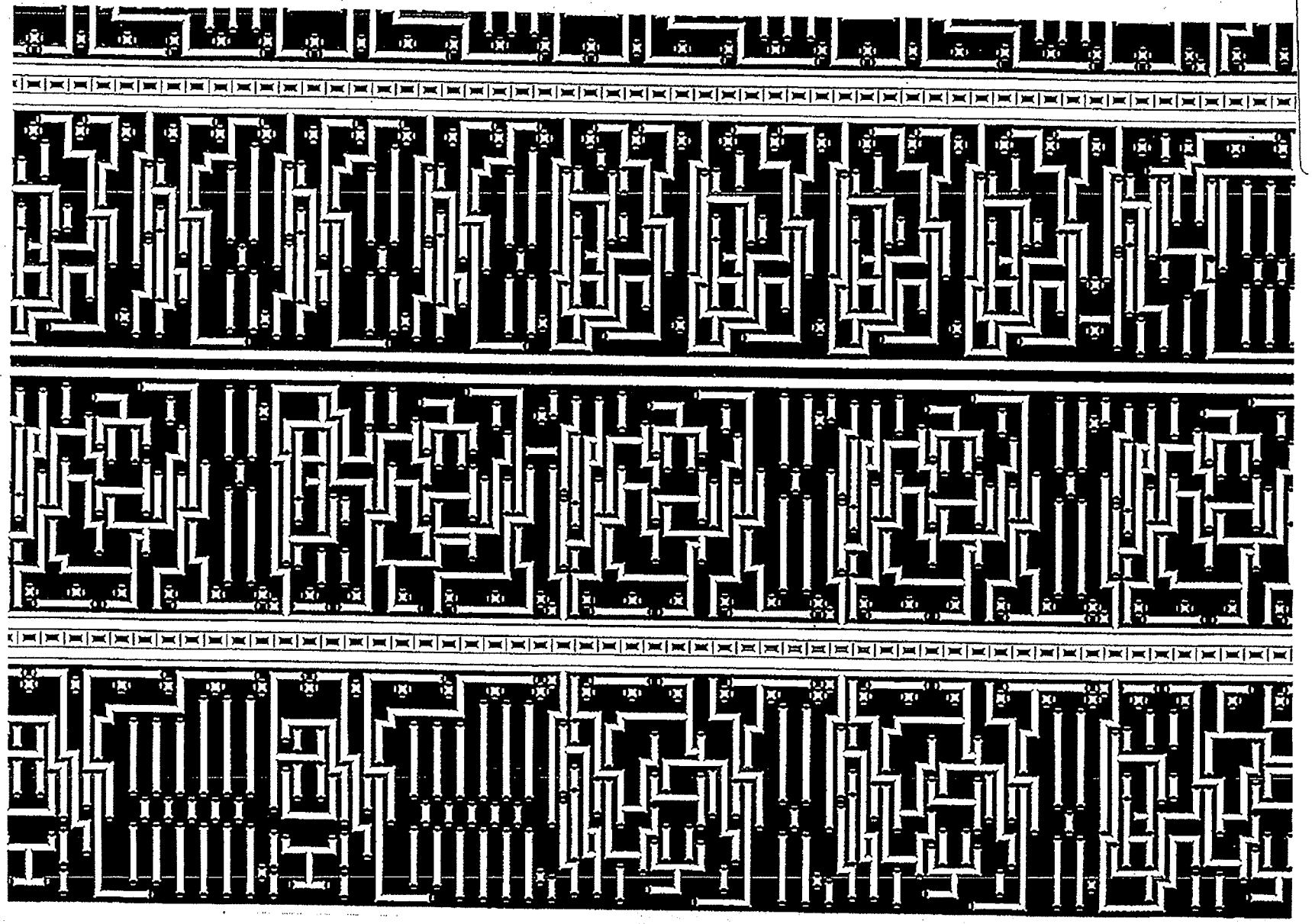
REDACTED

MICROUNITY CONFIDENTIAL



Pattern file: "199991800.03" Created:

REDACTED MICROUNITY CONFIDENTIAL



Pattern file: "19999 1800.03" Created: REDACTED

MICROUNITY CONFIDENTIAL

MU 0020450

CONFIDENTIAL

MU 0020451
CONFIDENTIAL
METALS SUPPORT PEDESTALS FOR METAL 4
ORIGINAL METALS
WIRING METALS
MICROUNITY CONFIDENTIAL